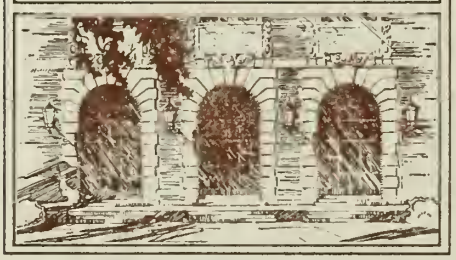
LIBRARY OF THE
UNIVERSITY OF ILLINOIS
AT URBANA-CHAMPAIGN

510.84
Il6r
no. 812-817
cop. 2



CENTRAL CIRCULATION BOOKSTACKS

The person charging this material is responsible for its renewal or its return to the library from which it was borrowed on or before the **Latest Date** stamped below. **You may be charged a minimum fee of $75.00 for each lost book.**

Theft, mutilation, and underlining of books are reasons for disciplinary action and may result in dismissal from the University.

TO RENEW CALL TELEPHONE CENTER, 333-8400

UNIVERSITY OF ILLINOIS LIBRARY AT URBANA-CHAMPAIGN

MAR 1 1 1998

When renewing by phone, write new due date below previous due date. L162

Il6r
no.815
Cop2

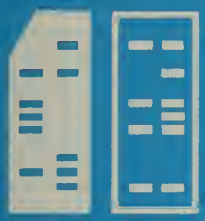
Report No. UIUCDCS-R-76-815

A TALKING COMPUTER TERMINAL

by

James Albert Kutsch, Jr.

July 13, 1976



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

The Library of the
NOV 8 1976
University of Illinois
at Urbana-Champaign

Report No. UIUCDCS-R-76-815

A TALKING COMPUTER TERMINAL

by

James Albert Kutsch, Jr.

July 13, 1976

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science in the Graduate College of the University of Illinois at Urbana-Champaign, 1976.

Digitized by the Internet Archive
in 2013

http://archive.org/details/talkingcomputert815kuts

ACKNOWLEDGMENTS

I would like to express my thanks to the following people for their help in this thesis: First, the late Professor Donald Gillies, for his early support and advice; Professor H. J. Poppelbaum and the members of my committee, for their guidance; The Graduate College, for their financial support of the project; and finally, my wife, Kimberly, for her many untiring hours of soldering hardware, library research, and proofreading, which were invaluable.

TABLE OF CONTENTS

1. A HISTORY OF READING AIDS FOR THE BLIND..... 1
1.1 Introduction 1
1.2 Braille 2
1.3 Image Converters 4
1.4 Michigan State University System 5
1.5 Reading Machines 6

2. PROBLEM SPECIFICATION 8
2.1 Man-to-Man Communications 8
2.2 Man-to-Machine Communications 10
2.3 Details of the Problem 11
2.4 Inadequacies of Previous Solutions 14
2.5 Analog Speech Synthesis Systems 16
2.6 Digital Speech Synthesis Systems 17
2.7 A Talking Terminal 20

3. SYSTEM DETAILS 23
3.1 Hardware Components 23
3.2 Software Description 27
3.3 Sample Sentence Being Processed 40

4. SUMMARY AND CONCLUSIONS 42
4.1 Summary 42
4.2 Evaluation 43
4.3 Hardware Improvements 45
4.4 Software Improvements 46
4.6 Conclusions 48

REFERENCES 49

APPENDIX I 52

VITA 89

CHAPTER 1

A HISTORY OF READING AIDS FOR THE BLIND

Section 1.1: Introduction

Finding a reading aid or method for the blind has been a long standing research problem. The earliest significant contribution to the problem was the work of Louis Braille in the mid 1800's. More recently, computers and other sophisticated electronic equipment have been used in an attempt to find a better solution.

Because of the specific nature of the problem, or perhaps because it affects only a small percentage of the population, it has not received the research attention it deserves. Some major contributions have been made, but they are few and far between. Further, almost all of the research that has been done was conducted by sighted individuals, with little or no input from blind persons who would be the eventual users of any discovery.

A sighted person's opinion of what he would want or need if he were blind is not always a true reflection of the needs of the blind. The author, himself blind, hopes that his personal experience will lend an insight into the

problems of the blind which is sometimes lacking in other research.

A discussion of significant previous research will follow. More emphasis will be given to computer aids for the blind, since it is the subject of this thesis.

Section 1.2: Braille

Braille is the oldest, most well-known method of reading for the blind. The Braille cell consists of six dots, organized in a matrix two dots wide and three dots high. All letters, numbers, and punctuation symbols are formed by some combination of one or more of these dots. For example, the letter 'a' consists of the upper left dot, 'b', the upper left and middle left dots, and 'c', the upper left and upper right dots.

The Braille dots are raised bumps on the paper. These bumps are commonly embossed by two methods. A Braille slate and stylus is a small portable device for writing Braille and the Perkins Brailleswriter is a larger, faster, more automatic model device. In both cases, a metal pin is pushed through the paper into a metal cup on the reverse side.

Commercially printed Braille, such as books or magazines, are produced with a special printing press which can emboss the Braille on both sides of the paper. It produces all characters on both sides simultaneously.

Braille can also be produced by computers. Several manufacturers offer a Braille print train for their line printers, but a less expensive software technique is commonly used by most blind programmers. Basically, a soft cushiony material is placed behind the paper on the printer. Software routines are employed to convert the characters to be brailled into the appropriate series of periods and spaces, such that, when printed, the periods will push through the paper into the cushion, making a raised dot on the reverse side of the paper.

Loeber from IBM Corp. proposed an on-line Braille computer terminal. [2] The terminal would provide both a Braille and an ink-print copy of the data. Further, it could be used in a local, as well as the on-line mode.

With this device, Loeber suggests that a blind computer programmer can check his work and can interact with a time sharing computer system. Also, it could be used by sighted persons, producing the Braille copy to be read later by a blind person.

This system requires software resident in the computer system to perform the translation to the codes necessary to control the Braille mechanism in the terminal. At the time of the writing of his article, the terminal was not under commercial production.

Section 1.3: Image Converters

Probably the oldest reading device is the OPTOPHONE, invented in 1912. [20] The Battelle Aural Reading Device and the Mauch Laboratories VISOTONER are more recent variations to the OPTOPHONE design.

With these devices, a line is scanned and a sound pattern is generated for each letter form. The tone pattern is arbitrary and needs to be learned. Reading rates with these devices are extremely low.

Another image converter is the OPTICON (Image to Tactile Converter). [23] This device is being manufactured and distributed by Telesensory Systems Inc. A small camera is passed over a line of print. A raised impression of the letter in view of the camera is presented by means of a small matrix of vibrating reeds. The printed page is read by moving the camera across the lines of print while feeling

the raised letters with one finger of the other hand.

The OPTICON is small, light weight, and battery operated. It can be used to read any printed material. Further, a logic reverse feature is provided which, when enabled, permits the OPTICON to respond to light letters on a dark background. This feature allows the device to read the face of a cathode ray tube computer display.

Section 1.4: The Michigan State University System

A special terminal for blind computer users is being developed at Michigan State University. [13][21] This terminal provides synthetic speech and CRT output, with keyboard input. Special software in the University's computer supports the operation of this terminal.

Currently, a text editor, an interpreter for a subset of FORTRAN, and some CAI lessons are available in an experimental form. Each of these facilities has been specially designed or modified for use with the blind through the special terminal. Included in this modification is the generation of the codes needed to drive the speech synthesis device.

Future plans for the terminal include modification of other system facilities and experimentation with a means of indicating the position of information on the page. A change of dialect technique has been proposed. For example, messages near the top of the page might be read in a high pitched voice, while those nearer to the bottom would be read in a lower pitch. Some comparable scheme would be used for horizontal positioning on the page.

The terminal has been used by blind computer science students at Michigan State University. Their evaluations will be used to make future improvements.

Section 1.5: Reading Machines

A reading machine is a device with an unrestricted vocabulary that converts printed text directly into speech. A character recognition module scans the page and transmits data to the computer, where the letters are identified. The letters are then sent through a phoneme translator and syntax analyzer. Finally, the synthesis module produces connected speech output.

The reading machine requires a large dictionary, generally stored on a high speed drum. Computation time runs high as stress and intonation are assigned by complex analysis of parts of speech and context. [4][12][20][24]

The Massachusetts Institute of Technology machine currently requires a 3,000 morph dictionary, and can read at the level of a fourth grade reader. Computation is done on a PDP 9 with a high speed drum for storage. [4]

The University of Connecticut, in conjunction with Haskins Laboratories, is also researching reading machines with an eye towards a pilot reading center to serve the University's blind students. Synthetic speech is being produced with a Honeywell DDP-224 and a hardware synthesizer designed by F. S. Cooper. A trained phonetic typist translates printed text into phonetic text, thus bypassing character recognition and translation to phoneme problems. Prosodic features are assigned by the software. [24]

CHAPTER 2

PROBLEM SPECIFICATION

Section 2.1: Man-to-Man Communications

Communications through a non-verbal medium is a difficult problem for a blind individual. Since the visual channel is not available, all printed and otherwise visual information must be converted to data suitable for one of the remaining senses. The most commonly chosen secondary channels are auditory or tactile. Both are seriously limited in bandwidth as compared with vision, tactile even more so than auditory.

Common solutions include transcription to Braille and the reading of printed material either face to face or via a tape recorder. These processes can be broken down into two distinct parts. First is the assimilation of the material by the person reading or transcribing it, and second is the conversion and presentation of the data through the alternative medium, i.e. speaking or writing it in Braille.

Even though many textbooks and novels are available on tape and records from the Library of Congress and Recordings for the Blind and in Braille from the American Printinghouse

for the Blind, there are still serious drawbacks to the loss of the visual channel. Of primary concern is the availability of this material. That is, although brailled and recorded books are adequate, they are not as readily available as the printed word. Not every book, paper, or magazine that a blind person might wish to read has been converted. Futher, the conversion is time-consuming and requires the assistance of a sighted person.

A major contribution to the solution of this problem was made by Telesensory Systems Inc. when they began production of the OPTICON (optical to tactile converter). [23] A small camera is moved across a line of printed information. A tactile image of each letter so scanned is presented through a matrix of vibrating reeds. These raised impressions are read (felt) by the user. The user of the OPTICON can, without assistance, read the printed page. It should be noted that it takes considerable training to recognize the raised impressions of the letters, and even then, reading rates remain disappointingly low. Experienced users have not been able to exceed 70 words per minute. [1]

Section 2.2: Man-to-Machine Communications

A blind computer user is faced with the same problems of communications except that he is communicating with a computer rather than another person. Obviously, the input to the computer is not a problem, since key punches and terminals are keyboard devices and although the location of the keys may vary, they are quite similar to those of a typewriter. Thus, input to the computer can be achieved with only the inconvenience of learning a new keyboard, which is also required of the sighted computer user.

The transferal of information from the computer to the blind user is similar to the man to man communications problem discussed above. However, there are two major differences. First, the data cannot already have been transcribed into Braille or recorded on tape because every computer output listing is unique. This means that the blind user must make arrangements to have someone transcribe or read his output or he must read it himself with an image to tactile converter. Futher, it is quite difficult, if not impossible, for the average blind person to use a time sharing terminal interactively without someone with him to read the output or without the use of an OPTICON.

The second difference is that the first of the two part conversion process, i.e. the reading, is eliminated since the data is already in computer readable form. If the computer could itself present the data in something other than a visual medium, the man to machine communications problem would be solved. Now, only part two, how to convert and present the data needs be determined.

Section 2.3: Details of the Problem

The purpose of this thesis is to design an optimal method for a blind person to communicate with the computer and to assemble whatever hardware and software is necessary to achieve such communications. The criteria against which the adequacy of a solution can be measured will be set forth so that any shortcomings of a particular system can be more easily discovered.

One of the most important qualifications of any communications technique is that it should be available whenever the individual wishes to use the computer system. This precludes the use of another human being to read the output, since scheduling can be quite difficult. Therefore, the communications method should involve some machine or

mechanism that is available whenever the computer itself is available to the average user.

Second, the communications method should be as complete and as versatile as possible. It should be able to be used in many aspects of computer communications. For example, if the computer system can be used in a batch mode through punch cards and listings, and in a time sharing mode from a terminal, the communications method should be applicable to both.

Third, the converted data presented to the user should be as thorough and complete as that presented to a sighted user. All system messages, job control language, log on preambles, etc. which are generated from normal use should be presented unmodified to the blind user. Further, all system resources should be available. This precludes the use of restricted or specifically modified languages, compilers, editors, etc. for use by the blind.

Fourth, the communications method should not require that the user acquire some special or unusual skill, e.g. Morse code or the ability to distinguish musical chords. The user of the system should be able to interact with the computer with no more training than is expected of a sighted user.

Fifth, in the same way a teletypewriter can be attached to many widely varied computers, the communications method should be computer independent. This would allow the blind computer user to communicate with many different computers without requiring totally different methods for different computers. This goal is considerably more difficult to realize, but is very important.

Finally, the communications method should be equivalent in cost to devices required by a sighted person to communicate with a computer. A company wishing to hire a blind person should not be expected to pay large sums of money for equipment to enable the blind person to do the same job as a sighted person who would need no special equipment.

These six criterion will be used to measure the effectiveness of various methods of communications. Input to the computer will not be discussed because it is easily achieved through keyboard devices which blind people can operate without any difficulty. Emphasis will be on ways to convert and present data through a non-visual medium.

Section 2.4: Inadequacies of Previous Solutions

As is obvious from the discussion in Chapter 1, Braille is the most common and perhaps the easiest form of output for the blind. However, it is not without disadvantages. The Braille character consists of a matrix of dots, two wide and three high. Counting horizontal and vertical spacing between characters, 40 Braille characters would be 120 characters wide and four lines high on a printer. Thus, what would appear on one line of 120 characters of print requires a maximum of twelve lines of dots and spaces in Braille. Even with the use of compression techniques for blanks and short lines, Braille is still very bulky. For example, a thirty volume encyclopedia would consist of 145 volumes of five inch thick books in Braille. Further, computer generated Braille requires some sort of change to the line printer, which means a time delay in most computer installations. Special forms jobs are seldom run more than a few times a day. Because of its bulk, Braille output is quite wasteful, especially if only a few lines of each listing are needed, as is commonly the case in debugging a program. Braille output does provide a good means of storing information for later use and providing a listing which can be studied many times. However, it is easily destroyed by placing many listings or other heavy objects on

tcp of it, causing the raised dots to be erased. Properly embossed Braille cn special paper has an average life of fifty readings. [2] (Embossing against a soft base does nct produce as well as when a metal die is used. The soft base limits dot height by allcwing the dct base to expand.)

On the surface, the OPTICON seems tc present a very satisfactory sclutiocn. It is ccmputer independent, readily available, reads all forms of output (it can even be used to read the face cf a cathcde ray tube display), and presents no restrictiors cn what computer services are used. It is, in fact, quite adequate for reading listings in a batch environment. But, since it presents the tactile data on the user's fingertip, it is inccnvenient to use at a time sharing terminal, since the user must continually move his hand frcm the OPTICON to the keyboard and back again. Also, the reading speed with the OPTICON is a consideration against its use, as stated earlier.

Additionally, with both Braille and the OPTICON, a sensory fatigue prcblem enters after prolcnged use without rest. This fatigue could be compared to eye strain. The Braille or CPTICCN reading finger becomes tired and the ability to detect and distinguish characters lessens, causing errors and slowdown in reading. Althcugh experienced readers may be able tc work for several hours

before becoming fatigued, others cannot continue for more than a half hour without giving the reading finger a rest. Since most computer programmers spend several hours at a time at a terminal, this drawback would present serious problems.

The final medium to discuss is that of synthetic speech. Many companies are producing various types of voice response systems. However, in most cases, system details are proprietary. For a recent survey of commercially available units see [5]. Synthetic speech seems to be the optimal medium for the computer to communicate with blind users, indeed, with man. There is sufficient reason to believe that no arbitrary letter by letter code could ever be as efficient and understandable as the spoken language of the user. To bridge the gap between the computer and the blind user, the computer must 'speak'. [3]-[4]

Section 2.5: Analog Speech Synthesis Systems

The earliest voice response systems were analog systems. Speech waveforms for words, phrases, and occasionally even syllables were recorded. Most commonly, photographic film or magnetic tapes were used to store the

sounds, as these could easily be mounted on drums and recorded in tracks. Drums rotate slowly and a reproduce head or heads move in place when a track is chosen. Because the drums are slow and large, duplicate entries are necessary around the surface to provide rapid access to a segment. Switching of heads is either controlled by a mini-computer or is hard wired.

These methods are very useful in systems that require a small fixed vocabulary, e.g. time and temperature units. However, their adaptation to more sophisticated systems is limited. First, there is little flexibility imposed by the mechanics of the device. Second, preparation of new or additional vocabularies is slow and expensive. Whole films must be reproduced. Third, sounds are far from natural since they are concatenations of isolated words or sounds. [5]-[6]

Section 2.6: Digital Speech Synthesis Systems

Recently, Masters Specialties Company has developed a technique for digitalizing and storing whole words in metal oxide semiconductor (MOS) read only memory (ROM) chips. By a complex plotting of waveforms, engineers have converted

analog audic signals into digital signals requiring a minimum of storage.

The advantage is first, a mcre natural sounding voice. All of the vccal qualities of the original speaker are there. Each word is stcred in its own memory which eliminates the need fcr ccstly drums or disks. Accessing is simplified by directly addressing the ROM instead of ccntrclled access by a mini-computer or hard wiring, since logic decoding is acccmplished withcut ccmplex programming.

The EVA (Expandable Voice Annunciator) is capable of a vocabulary cf cnly ten to thirty words. Cnce again, while an improvement over the analog devices, this system would nct be suited tc any large vocabulary needs.

Rather than dealing with large units such as phrases or words, the most versatile synthesizers deal with phonemes, the smallest unit of the spoken language. They accept digital phcneme ccmmands and return their audible ccunterpart. This approach requires sophisticated software to convert from letters to phonemes. There are several algorithms fcr this translation ranging from dictionary look-up techniques [4][11]-[12] to synthesis by rule. [7]-[10]

The problem with this method of synthesis is that the words to be synthesized must be converted from graphemes to the corresponding phonemes. (Graphemes are the 26 letters of the alphabet and all special symbols.) This is not a trivial process since the English language does not follow any simple set of rules for letter to sound correspondence. This is most apparent with vowels. For example, the sound corresponding to the grapheme 'O' in 'women' and 'Bob' is clearly quite different. Since phoneme translation is context dependent, these algorithms usually require large tables and a moderate amount of computation. However, computer generated speech systems using synthesis by rule have been designed to produce quite satisfactory and understandable output. [10][13]

An example of such a system is the one at Michigan State University described in Chapter 1. This system is a very significant step towards an optimal communications method between a computer and a blind user. Nevertheless, there are some serious shortcomings. First, the translation to phonemes is done in the host computer system, not in the terminal. Thus, the system is computer dependent. Although the software might be moved to another computer, such a transfer would almost certainly involve modification to the software, perhaps major revisions if the transfer were to another model or another manufacturer's computer. Further,

only a subset of the system's resources are available to users of the special terminal. This may be adequate in beginning computer science education but is not at all satisfactory for a more advanced and experienced computer user. The most serious drawback, however, is that since the conversion to phonemes is a process on the host computer, some messages can be sent directly to the terminal, bypassing the translation process. These characters, if sent directly to the speech synthesis device, would not produce words, and if sent to a teleprinter or CRT for printing, they could not be read by the user.

Section 2.7: A Talking Terminal

Considering all aspects of the above discussion, it seems that an optimal communications system for blind computer users could be designed. Such a system should clearly be an interactive terminal, rather than a batch system, to allow more complete access to computer facilities. All batch facilities are usually available through an interactive terminal, but seldom is the reverse true. That is, in a terminal system, long jobs can be scheduled for execution in batch mode and have the output sent back to the terminal for later reading. However,

interactive computing, and especially text editing, is available in a batch environment only in a very limited way.

Interactive text editors are of great advantage to the blind user. Corrections to program files can be made easily by reference to line numbers without the inconvenience and difficulty of finding specific punched cards in a deck.

Many sighted programmers use on line terminals to write and debug their programs. This two way communication speeds up the process. However, the blind programmer does not have this capability. Clearly, this disadvantage is felt. Recently, the ACM Special Interest Group for Computers and the Physically Handicapped circulated a questionaire among blind programmers about their special needs and interests. The results were published in SIGCAPH Newsletter Number 8, July 1, 1973. In response to 'What special tools or equipment would you like to see developed?', readers wrote: 'A faster means of reading than Braille', 'A machine to read ink output from the computer.', 'Auditory output or input echo', 'Random access books', 'An easier way of finding or inserting cards in a deck', 'A card reader enabling me to make corrections myself'.

Computer generated speech of the phoneme synthesis variety is the best means of presenting the output. However, the translation from graphemes to phonemes clearly

should not be done in the host computer due to the computer dependency problem and the inability to translate certain messages as mentioned above. Therefore, an intelligent terminal of some description must be incorporated so that the translation can be done in the terminal. This solves both problems. Since all messsages are sent to the terminal and are translated there for presentation to the synthesizer, no messages can bypass proper translation. Also, the terminal can be constructed in such a way that it appears to be a teletypewriter or other conventional terminal to the host computer system. Thus, computer independence is maintained.

Additional advantages are present which can be attributed to the intelligence of the terminal. That is, some special processing of the data presented to the terminal can be performed if desired, e.g. reading only the first few characters of a line, selecting a mode of operation whereby the terminal spells all output, and other local features.

It appears that such a talking computer terminal would be the optimal method of communications for blind computer users. Specific details of the talking terminal will follow in Chapter 3.

CHAPTER 3

SYSTEM DETAILS

Section 3.1: Hardware Components

An analysis of a teletypewriter or a CRT type terminal yields the following components: a keyboard to enter the data; a coupling device such as a modem to communicate with the computer; and an output device, such as a teleprinter or cathode ray tube display. The talking terminal retains the keyboard and coupling device, and adds a speech synthesis device for output presentation, driven by a micro-processor. The micro-processor is necessary to maintain host independence, since synthesis devices need special codes to drive them. The standard ASCII characters sent to a terminal need to be converted to these special codes.

In the selection of a specific manufacturer and model of the above equipment, several factors were taken into consideration. The talking terminal is considered to be a prototype device for experimentation. Therefore, ease in entering, modifying, and debugging the system's software was of great importance. In a final version of the terminal

such features as front panel display lights and control switches, which were essential in the system's development, would not be necessary, and, in fact, might be undesirable. Other considerations in hardware selection were keeping the cost of the terminal as close as possible to the cost of a standard terminal, within the constraints of delivery time and availability of components.

The Votrax unit, manufactured by the Vocal Interface Division of Federal Screw Works was selected for the speech synthesis device. This decision was based on several factors. First, the majority of speech synthesis applications found in the literature make use of the Votrax, and therefore, many of the algorithms for translation from text to phoneme codes are designed specifically for use with the Votrax. Further, after hearing the device demonstrated on several occasions, it was deemed adequate for the purposes of the talking terminal.

Another factor was that the Votrax is driven by phonetic codes, which means little dictionary storage is necessary. In contrast, examine the dictionary storage requirements for formant synthesis. A typical desk size abridged dictionary contains 130,000 words. This must be expanded to include pronunciation of the common endings of most words, plurals for nouns, past tense for verbs,

suffixes, and prefixes. Also, many entries may never be needed. In short, a pronouncing dictionary would require around 500 million bits of storage. [14] A more compact method is obviously imperative and phonetic transcription is one answer.

The Votrax unit is an electronic analog of the human vocal system. It accepts as input, a series of eight bit digital command words, indicating the phonemes desired. (The phoneme is the basic building block of speech.) The audio output, which is an electronic simulation of human speech, is compatible with any conventional audio circuit.

The Votrax speech synthesizer consists of three printed circuit boards, which contain a series of active filters to mimick the resonance in the vocal and nasal tracts. Special forcing function generators provide the filter excitation synthesizing the vocal chords and fricative sound sources. Unique low frequency function generators supply the articulation and dynamics used when transitioning from one phoneme to another. Perfect pronunciation and articulation for each phoneme is accessed from a read only memory in the device. [15]

The micro-processor system was selected to provide the intelligence for the terminal. Due to the slow data rate of speech, (on average, two to three words per second), the

speed of the computer was not an important factor, even with a complicated algorithm for text to phoneme translation.

The Altair 8800 computer system manufactured by MITS, Inc. was used in the talking terminal. This decision was based on availability, low cost, modular design, and a bus structure that allows the system to be configured with any amount of memory and any number of input/output ports. The availability of several Altair compatible boards from other manufacturers at low cost enhanced the list of advantages for the Altair 8800.

The complete central processing unit (CPU) is on a single LSI chip, Intel's 8080 micro-processor. This chip has a separate sixteen line address and eight line bidirectional data bus. The transfer of data between the CPU and memory or I/O takes place in a 100 line bus structure. These also carry the unregulated supply voltages, and all control and status signals. Other cards in the system can control the bus only when addressed by the CPU. [16]

Input/output drivers and 16K bytes of static random access read-write memory were obtained from Processor Technology to complete the Altair computer system. Read-write memory was selected for the protoype terminal so that program changes could be easily effected. A final

version of the terminal would require only 2K bytes of read-write memory for buffers and program variables. The remaining 14K bytes could be replaced with read only memory, permitting non-volatile storage of system software and tables.

To complete the terminal, a ComData model 150-12 modem and a Cherry Switches keyboard were used. However, any EIA compatible modem and any eight bit parallel ASCII keyboard could be substituted with no modification to the terminal.

The complete talking terminal, consisting of four units, can easily fit on top of a small table. In fact, since only the keyboard, modem, and some type of speaker need be within arm's reach of the user, the terminal can effectively be made very small by placing the other components in an out of the way place.

Section 3.2: Software Description

The software for the terminal consists primarily of routines to perform the translation from English text to the digital commands necessary to drive the speech synthesis device. Routines to handle the communications between the terminal and the host computer system, as well as those to

manage character storage buffers, complete the list of the terminal's software. The algorithms for the text to phoneme translation used in the terminal are those described by McIlroy from Bell Laboratories. [10] However, the algorithms have been modified to better suit the needs of the talking terminal.

The decision to use McIlroy's algorithms was based on several factors. First, his approach was designed specifically for use with the Votrax synthesizer. His rules and tables map directly into the phonemes used by the Votrax. (The phonemes used by the Votrax are not in a one to one correspondence with those in the International Phonetic Alphabet.) Further, McIlroy's algorithms deal with each word individually. That is, no attention is paid to context during the translation to phonemes. The talking terminal is designed to be used mostly, if not primarily, with computer languages. Abbreviated system messages of the type frequently found on the average time sharing system and other output data are usually not complete sentences. Therefore, the more expedient, less complicated method of independent processing of each word is better suited to this application than a context dependent process.

Allen and Lee have reported on the research at Massachusetts Institute of Technology aimed at a direct text translation. [4][11]-[12][17] First, whole words are isolated and looked-up in a translation table. If more than one transcription exists, a parts of speech preprocessor is invoked to establish which transcription should be used. Stress analysis is performed and the resultant data is sent to the speech synthesizer parameterization process. If the word is not found in the table, it is broken down into its constituent prefixes, suffixes, and word stems, and the process is repeated for each. If this is still unsuccessful, a set of letter to sound rules are applied in an attempt to approximate the correct pronunciation.

This approach is not well suited to the talking terminal application. It is designed to be used as a reading machine for the blind. Hardware requirements alone make it very undesirable for the talking terminal. The system is being implemented on a PDP 9 and a TX-0 computer and requires a high speed drum for dictionary storage. The dictionary currently consists of 11,000 words and approximately 400 letter to sound rules. [18]

The University of Keele system [19] is also undesirable for use in the talking terminal. This system relies heavily on breath group and stress analysis and the rules are

designed to produce synthetic speech in a British dialect. Context dependent algorithms as previously mentioned, especially those with parts of speech preprocessors, are not suited to the terminal's usage. Complete naturalness of synthetic speech as an end in itself is not the goal of the talking terminal, but rather, understandable output, readily available to the blind programmer. The hardware and software requirements of McIlroy's algorithm are reasonable and they provide a fast, reasonably accurate translation which is quite understandable to most users, once they become familiar with the machine's accent.

The main body of the software is a 10,000 byte table which contains the letters of the alphabet, special symbols, some exception words, and word fragments, all with their corresponding phoneme equivalents. Letter markers (an asterisk) preceed letters and special symbols while fragment markers (a percent sign) preceed the word fragment in the table. This is needed to distinguish cases where a letter may appear more than once, e.g. '*a' is the letter 'a', pronounced 'ay'; 'a' is the word 'a', pronounced 'uh', and '%a' is the fragment for a short 'a'. In some cases, replacement fragments are included in the table with the word fragments. These are used to replace a given word fragment during processing. For example, 'alk' is replaced with 'awk' in words like 'walk' and 'talk'.

The software consists of routines to access this table and preprocessing routines which mark long and short vowels, pronounced or silent 'e's , etc. Briefly, a word is processed as follows. First, the table is searched to see if the word is an exception. That is, a word which defies processing by rule and which has been included in the table along with the phonemes necessary to properly pronounce it. If so, processing is complete. If not, and if the word is plural, the final 's' is removed. The exception table is again searched for this singular form. If this search still fails, the preprocessing rules are applied and the table is searched repetitively for fragments. The phonemes corresponding to these fragments are placed in the output buffer and any replacement letters are added to the beginning of the remaining fragment. This fragment search continues until the entire word has been processed or until the table search fails to find a fragment. If the attempt to translate the word by rule fails, the word is translated into the phoremes necessary to spell the word, letter by letter. Spelling is a last resort and is usually only invoked with unusual abbreviations and the like.

All software was written in 8080 assembly language and appears in Appendix I. Detailed descriptions of each routine in the program follow.

All processing begins with the interrupt service routine, 'INT'. The hardware is configured such that an interrupt occurs whenever a key is pressed on the keyboard or a character is received by the modem. In the case of a keyboard interrupt, the key stroke data is read and checked to see if it is a 'control d' (octal 004). This key indicates that the terminal should discontinue reading the current line and proceed to the next line. This feature is analogous to a sighted user scanning only the first few words on a line and skipping to the next one.

If the key stroke was the discontinue key, a flag is set which is tested in a subsequent routine. Otherwise, the key stroke data is transmitted through the modem to the host computer, ending the processing of the input interrupt.

Processing a modem receiver interrupt is somewhat more complicated. When this interrupt occurs, the received data character is read from the modem and placed in a circular character queue, assuming that the queue is not full. If it is full, this character, and all subsequent characters are ignored until space is again available in the queue. A warning tone is turned on to indicate to the user that characters have been lost.

Since communications with the host computer are through an asynchronous data line, the terminal itself cannot request that the host computer wait a while until buffer space is available. This responsibility is left to the user of the terminal. The queue in the current implementation of the terminal will allow up to 1024 characters. The characters are, of course, removed as the words are spoken by the Synthesizer. It is only when the host computer gets significantly ahead of the terminal that problems occur. A careful user can easily see that the buffer overflow problem never occurs. He need only wait between requests to the host computer long enough for the terminal to catch up in reading the output.

The routine 'SPEAK' is what might be considered the main routine of the entire program. It directs the execution of all other routines. First, it repetitively calls 'GET', each time getting a character from the input queue, until an end of line marker is returned. As each character is fetched, it is stored in a temporary buffer with all words, numbers, and special symbols separated by a mark character (octal 000). After an entire line has been reformatted in this manner, a loop is entered that removes each word, number, or special character, item by item from the reformatted line. The item is first passed to 'PHREAD' to see if it exists in its entirety in the exception table.

If it does, 'PHREAD' returns the corresponding phoneme equivalent for the item. If the item is not found in the exception table, 'PHPRON' is called to try automatic pronunciation by rule. If the automatic pronunciation suceeds in translating the item to its corresponding phonemes, the phonemes are again returned. If unsuccessful, i.e. if the item cannot be proncunced because it has no vowels, consists of only one letter, etc., 'PHSPELL' is called which returns the phonemes necessary to spell the word. 'PHSPELL' is called directly if the 'spell only' switch has been set. Next, the phonemes are passed to the speech synthesizer by the routine 'VOTRAX'. After all items in the line have been processed, 'SPEAK' checks for other lines to be read.

As mentioned above, 'PHREAD' checks to see if the entire word is in the exception table. This is accomplished by passing the word to 'PREFIX'. If 'PREFIX' finds the entire word in the table, rather than only a prefix of the item, 'PHREAD' copies the phonemes from the table into the output buffer and returns. If 'PREFIX' does not find the entire word, 'PHREAD' returns and indicates that the word is not in the exception table.

The routine 'PREFIX' is passed an item to be looked up in the table. It finds the longest possible prefix of the item. The address of this prefix is returned to the calling routine. In addition, a flag indicating failure to find any prefix, success in finding a prefix, or success in finding the entire word is returned to the calling routine.

'PHPRON' parses the word and attempts to pronounce it by certain rules found in the word fragment table. First, 'FINALS' is called to check for and remove the final 's' of the word. Also, final 'ie' is changed to 'y'. If a final 's' was removed, 'PHREAD' is again called to determine if the singular form of the word is in the exception table. If it was found, the phoneme for 's' is added at the end of the output buffer, and 'PHPRON' returns indicating success in finding the phoneme equivalent for the word.

If a final 's' is not found, or if the singular form of the word is still not in the exception table, preprocessing routines are called to find and mark long vowels, silent 'e's, standard suffixes, and voiced or unvoiced 's's. These routines are 'FINALE', 'MIDU', 'MIDE', and 'MIDS'.

After this preprocessing is completed, any previously removed final 's' is replaced and end of word markers are placed at the beginning and end of the word. This is done so that subsequent table searches for prefixes can return

different phonemes for certain letter combinations when found at the beginning, in the middle, or at the end of a word. For example, the final 'a' in 'algebra' and 'India' is pronounced 'uh' but the 'a' found in the middle of 'cat' is pronounced short.

A loop to find prefixes is now entered. A word fragment marker is placed at the beginning of the letters that remain in the word, and 'PREFIX' is called to find the longest possible prefix of the letters. If during any iteration no prefix is found, automatic pronunciation by rule fails and 'PHPRON' returns. When a prefix is found, it is removed from the word fragment, the phoneme equivalent for the prefix is written into the output buffer, and any replacement letters which accompanied the rule are written in front of the remaining fragment. Then, the next iteration is begun. This continues until all of the letters of the word have been removed, in which case the word has been translated into its phoneme equivalent, or until a prefix can no longer be found, indicating failure.

'PHSPELL' breaks the word into letters. Each letter, with the letter marker in front of it, is then passed to 'PHREAD' to find the phoneme equivalent. All letters of the alphabet and special symbols are in the exception table with their corresponding phonemes. After all letters have been

processed, the cutput buffer containing the phonemes is returned. 'PHSPELL' cannot fail. That is, it is a last resort after table lcok-up and automatic pronunciatien by rule have both failed, and always results in the phonemes necessary tc spell the word. However, certain unprintable terminal and ccmmunicatiers control characters are not in the table and are therefore ignored if encountered in the spelling process.

As described above, the final 's' preprocessor, 'FINALS', checks to see if the last letter in the word is an 's'. If a final 's' is fcund and the preceeding letter is an 's' or a 'u', i.e. the word ends in 'ss' or 'us', the 's' is not removed since it does not usually indicate plural form. In this case, final 's' processing is terminated.

After removal cf a final 's', the routine tests to see if the 's' should be a voiced 's' ('z' sound) or an unvoiced 's' ('s' sound). If the final 's' is preceeded by one of 'c', 'f','k', 'p', or 't', then the 's' is unvoiced. Ctherwise it is voiced.

A final 'ie' is changed to 'y' regardless of the presence of a final 's'. On exit from the routine, a flag is returned indicating that nc final 's', a voiced final 's', or an unvoiced final 's' was fcund.

The final 'e' preprocessor, 'FINALE', marks silent final 'e's, some instances of long vowels, and standard suffixes. The first test determines if the last letter of the word is an 'e'. If so, and if it is the only vowel in the word, it is marked long by capitalizing it, e.g. 'he', 'she'. The practice of marking long vowels by capitalizing them will be observed throughout the program. If the above test succeeds, final 'e' processing is complete since there are no other vowels in the word.

By means of repetitive calls to 'SUFFIX' and 'INSERT', all standard suffixes at the end of the word are found and separated from each other and from the word stem by the insertion of a break mark (the question mark). Subsequent processing deals mainly with the word stem.

If the last letter of the word stem is an 'i', 'u', or 'y', and if it is the only vowel in the stem, it is marked as a long vowel. Again, since no vowels remain in the stem, processing is complete.

After this processing is complete, certain instances of long vowels and voiced or unvoiced 'th's are located and marked. This completes final 'e' processing.

'MIDU' performs three functions. First, it locates and marks all long 'u's. Second, certain instances of 'a', 'e', and 'o' are marked long if they meet the necessary specifications. Finally, a test is made to see if the first vowel of the word is an 'i' or 'y' (except when the 'y' is the first letter of the word). If so, the 'i' or 'y' is marked long.

Medial 's' processing, done by 'MIDS', locates all 's's except in the first and last letter of the word. If the 's' is a voiced 's', i.e. is to be pronounced as a 'z', it is marked so by capitalization. The rule governing this decision is: an 's' is marked voiced if it is preceeded by a vowel and followed by a vowel or the letter 'm'.

The other preprocessing routine which remains to be discussed is 'MIDE'. This routine locates silent medial 'e's and marks them as silent by insertion of the break mark. Medial silent 'e's primarily occur in compound words whose first part ends in 'e', e.g. 'houseboat' and 'bumblebee'.

To facilitate frequently used tests, the following subroutines were also included in the software: 'ONEOF', which determines if a letter is an element of an indicated set of letters; 'VOWEL', which determines if a specified fragment of a word contains any vowels; 'SYLTEST', which

determines if a word fragment is a syllable; and 'TH', which determines if the next two letters at a specified location are the letters 'th'.

Section 3.3: Sample Sentence Being Processed

To help demonstrate the software's operation, the sample sentence ,'PLEASE LOG IN:', will be followed through the routines.

First, each character is obtained from the character storage queue. As they are fetched, extra blanks are removed and words, numbers, and special symbols are separated from each other by insertion of a mark character (binary 0). After this processing, the work buffer contains: 'please 0 log 0 in 0 : 0 0'. Notice that all letters have been set to lower case. This is done by the interrupt service routine and is necessary since capitalization is used to indicate certain pronunciation features. The double mark characters are used to indicate end of the line.

Next, 'please' is looked up in the exception table. Since it is not an exception word, this search fails and automatic pronunciation is attempted. No final 's' exists,

so the preprocessing routines are called next. The final 'e' is marked silent by the insertion of the break mark (question mark), and the 's' is marked as a voiced 's' by capitalization. After preprocessing, the buffer contains: 'pleaSe?'.

Now, beginning and end of word marks are inserted and the fragment mark is inserted in front of the word, resulting in: '%#pleaSe?#'. Repetitive searches in the rule table are then made for prefixes. These searches result in the following phonemes and replacement characters:

FRAGMENT	PREFIX	PHONEMES
%#pleaSe?#	%#	
%pleaSe?#	%p	p
%leaSe?#	%l	l
%eaSe?#	%ea	ie, y
%Se?#	%S	z
%e?#	%e?	
%#	%#	

As seen above, the word 'please' results in the phonemes: 'p, l, ie, y, z'. Similar processing takes place for the words 'log' and 'in', in the input sentence. The colon in the input fails automatic pronunciation and enters the spelling routine. However, after the letter mark is placed in front of it, it is found in the exception table, resulting in the phonemes: 'k, c, l, uh, n'.

CHAPTER 4

SUMMARY AND CONCLUSIONS

Section 4.1: Summary

In an attempt to provide a communications medium for blind computer users, the requirements for an optimal method of communications were set forth. Previous solutions to the problem were studied and their shortcomings were noted. Through a combination of ideas from previous approaches and the author's personal experience, a new and unique direction was taken, which resulted in a host independent talking computer terminal.

The hardware for the terminal, consisting of a micro-processor, a speech synthesizer, a keyboard, and a modem, were assembled to form a prototype system. An analysis of algorithms for the translation from text to synthesizer commands was conducted, resulting is a decision to use a technique designed by McIlroy from Bell Laboratories, with modifications to tailor it to the needs of the talking terminal. The necessary software was written and the talking terminal was demonstrated using the University of Illinois' PDP 10 as the host computer.

Section 4.2: Evaluation

With respect to the six criteria for an optimal communications method between a computer and a blind user as set forth in Chapter 2, the talking terminal is quite adequate. It is certainly available any time that the computer system is available and a blind user wishes to use it.

The versatility criterion is met as well as it is by any type of terminal. Since most batch systems have a facility for the scheduling of jobs and retention of output for later review on a terminal, the user of the talking terminal would be able to operate both in batch and time sharing mode. It is recognized that the terminal would be of no use in a strictly batch environment. However, Braille listings, which are easily available, would help fill this need.

All processing necessary in the translation from text to the control codes necessary for the speech synthesizer is done in the terminal. In fact, to the host computer system, the terminal looks like any other ASCII asynchronous dial-up terminal, e.g. teletypewriter, CRT, etc. The necessity for and advantage of this approach is that all messages routed to the terminal from any source are translated and spoken to

the user by the synthesizer. There is never a case where a message can be sent to a user without the proper processing. Futher, host computer independence is maintained. The terminal can be used without change, either to itself or the host, on any computer system which allows ASCII asynchronous communications.

Clearly, since the talking terminal communicates with the user in spoken English, there is no requirement of learning a new skill. A user does, of course, need to learn the key locations on the keyboard and how to control the terminal. This requirement would be applicable to any type of terminal. Additionally, however, some users might need a short time to become familar with the terminal's speech. The need for practice time varies greatly from user to user, but is almost always less than fifteen minutes to a half hour.

The final criterion is that of cost. The prototype talking terminal was built for a total cost under $2,500. This is only slightly higher than $2,000, which is the average cost of conventional teletypewriter or CRT type terminals. A final version of the talking terminal, without software development and debugging requirements could be realized, it is estimated, at a cost less than or at worst, equal to the $2,000 figure given above.

Although there are no major shortcomings of the talking terminal, there are some changes which could improve a subsequent version of the terminal.

Section 4.3: Hardware Improvements

Hardware improvements consist mainly of replacing the micro-processor system. A smaller, less expensive micro-processor system, still based on the 8080 chip, is envisioned for the final version of the talking terminal. Features such as front panel display lights and control switches should be eliminated, as they would not be needed. Also , the random access memory should be replaced with memory of the read only variety. All system programs and tables would be permanently stored in this memory, and only a small amount of read-write memory would be needed for variables and buffers.

With this smaller design, it would become possible to mount all components of the terminal in one attractive reasonably sized case. The terminal would then become more portable and versatile.

All other system hardware for the talking terminal proved to be quite adequate and no other hardware changes would be made in subsequent versions.

Section 4.4: Software Improvements

No improvements are needed to the software, but the addition of more local control options might enhance the talking terminal's performance. The options to discontinue the reading of a line, and to force the terminal to spell all the words in a line already exist and have proved quite useful. It would also be useful if the terminal could repeat the last line read, without asking the host system to transmit the data again. Also, the ability to have the terminal read only certain columns of the output, e.g. only the first ten characters of each line, might be useful in some applications. However, this option might prove dangerous if an unexpected system message appeared and was not read in its entirety.

Neither of these improvements are significant and were not implemented in the prototype version in an effort to simplify operation of the terminal.

Much research has been undertaken to design algorithms for the assignment of stress within words and phrases. [11][18][22] Some simple rules for stress placement are quite successful, although seriously limited. For example, applying falling intonation before a period or comma, and rising intonation before a question mark have limited success. [18] Algorithms for more complicated and complete assignment of stress are still experimental. Computation of this type of stress assignment involves determining the part of speech of the word, as well as analysis of the word's location in the sentence or phrase.

Although the Votrax synthesizer is capable of pronouncing each phoneme at any of four levels of intonation, the talking terminal does not make use of this feature. As previously mentioned, perfect diction was not the main goal of the talking terminal. Since no short and fast method of determining stress placement was found, and the addition of more memory and use of more computation time were, at best, a tradeoff for only marginally better speech, this feature was not included. This is clearly an area which needs much future research.

Section 4.5: Conclusions

The talking terminal does provide an optimal means for a blind programmer to communicate with the computer. However, the applications to the sighted world should not go without mention. For instance, there are security systems where an individual must constantly monitor a CRT screen for messages. This tedious task could be eliminated by using the talking terminal in place of the silent CRT. The employee is then freed to do other tasks and still not miss any incoming messages.

The terminal has potential in computer aided instruction. Young children have conversation ability developments that far exceed their reading skills. The talking terminal could communicate with these children more extensively than the printed page. Also, in a classroom environment, the talking terminal could be coupled to a public address system allowing everyone in the room to hear the computer's output. Thus, one talking terminal would do the job of many conventional terminals.

It is hoped that the design and construction of the prototype talking computer terminal will enable blind programmers to be more self-sufficient and productive and will find many applications among sighted users.

REFERENCES

[1] J. C. Bliss, M. H. Catcher, C. H. Rogers, and R. P. Shepard, 'Optical-to-tactile image conversion for the blind', IEEE Trans. Man-Mach. Syst., vol. MMS-11, pp. 58-65, March 1970.

[2] N. C. Lceber, 'Proposed Braille computer terminal offers expanded world to the blind', Proc. AFIPS Fall Joint Comp. Conf., vol. 39, pp. 79-87, 1971

[3] A. M. Liberman, F. S. Cooper, D. P. Shankweiler, and M. Studdert-Kennedy, 'Perception of the speech code', Psychol. Rev., vol. 74, pp. 432-461, 1967.

[4] F. F. Lee, 'Reading Machine: from text to speech', IEEE Trans. on Audio and Electroacoustics, vol AU-17, No. 4, pp. 275-282, Dec. 1969.

[5] T. G. Homsby Jr., 'Voice response systems', Modern Data, pp. 46-50, Nov. 1972.

[6] L. R. Rabiner, R. W. Schafer, 'Digital techniques for computer voice response: implementations and applications', Proc. IEEE, vol. 64, , No. 4, pp. 416-433, Apr. 1976.

[7] L. J. Gerstman and J. L. Kelly, 'An artificial talker driven from phonetic input', Journal of Acoustical Society of America, vol. 33, pp. 835(A), 1961.

[8] J. N. Holmes, I. G. Mattingly, J. N. Shearme, 'Speech synthesis by rule', Language and Speech, vol. 7, Part 3, pp. 127-143, July-Sept. 1964.

[9] I. G. Mattingly, 'Synthesis by rule of prosodic features', Language and Speech, vol. 9, Part 1, pp. 1-13, Jan-Mar. 1966.

[10] M. D. McIlroy, 'Synthetic English speech by rule', Bell Telephone Laboratories, Mar. 1974.

[11] J. Allen, 'Machine to man communication by speech, part II: synthesis of prosodic features of speech by rule', Proc AFIPS. Spring Joint Comp. Conf.,vol. 32, pp. 339-344, 1968.

[12] J. Allen, 'Reading machines for the blind: the technical problems and the methods adopted for their solution', IEEE Trans. Audio and Electroacoustics, vol. AU-21, No. 3, pp. 259-264, June 1973.

[13] M. A. Rahimi and J. B. Eulenberg, 'Modes of information presentation for the blind programmer', Proc. Assoc. for Comp. Mach. Annual Conf., 1974.

[14] J. L. Flanagan, C. H. Coker, L. R. Rabiner, R. W. Schafer, N. Umeda, 'Synthetic voices for computers', IEEE Spectrum, pp. 22-45, Oct. 1970.

[15] Votrax Audio Response System Operators Manual, Vocal Interface Div., Federal Screw Works.

[16] ALTAIR 8800 THEORY OF OPERATIONS MANUAL, MITS Corp.

[17] J. Allen, 'Speech synthesis from unrestricted text', IEEE International Convention Digest, pp. 108-109, 1971.

[18] H. S. Elovitz, R. W. Johnson, A. McHugh, J. E. Shore, 'Automatic translation of English text to phonetics by means of letter-to-sound rules', Naval Research Laboratory Report 7948, Jan. 1976.

[19] W. A. Ainsworth, 'A system for converting English text into speech', IEEE Trans. Audio and Electro-acoustics, vol. AU-21, No. 3, pp. 288-290, June 1973.

[20] F. S. Cooper, J. H. Gaitenby, I. G. Mattingly, and N. Umeda, 'Reading aids for the blind: a special case of machine-to-man communication', IEEE Trans. Audio and Electroacoustics, vol. AU-17, No. 4, pp. 266-269, Dec. 1969.

[21] M. A. Rahimi and J. B. Eulenberg, 'A computing environment for the blind', Proc. AFIPS Natl. Comp. Conf., vol. 43, pp. 121-124, 1974.

[22] F. F. Lee, 'Machine-to-man communications by speech: part I: generation of segmental phonemes from text', Proc. AFIPS Sprint Joint Comp. Conf., vol. 32, pp. 333-3381968.

[23] J. C. Bliss, 'A relatively high-resolution reading aid for the blind', IEEE Trans. Man Machine

Systems, vol. MMS-10, pp. 1-9, Mar. 1969.

[24] P. W. Nye, J. D. Hankins, T. Rand, I. G. Mattingly, and F. S. Cooper, 'A plan for the field evaluation of an automated reading system for the blind', IEEE Trans. Audio and Electroacoustics, vol. AU-21, No. 3, pp. 265-268, June 1973.

APPENDIX I

PROGRAM LISTING

```
        .ENABL  LC
KB=0
VDATA=0
OSC=1
VCNTRL=1
STATUS=2
MODEM=3
TBE=200
RDA=100
VBUSY=40
KBDA=20
PE=4
FE=2
OE=1
B300=10
B110=4
VOUT=100
VIN=200
;
        .SBTTL  INTERRUPT SERVICE ROUTINES
;
;  THE FOLLOWING IS CALLED FOR EACH KEYBOARD OR MODEM INTERRUPT.
;
        .=73    ;THIS MUST APPEAR AT LOCATION 73
IINT:   PUSH    PSW             ;SAVE STATUS
        IN      STATUS          ;GET INT STATUS WORD
        PUSH    PSW             ;SAVE IT
        ANI     RDA             ;WAS INT FROM MODEM?
        CNZ     IIRDA           ;YES, HANDLE IT
        POP     PSW             ;GET INT STATUS BACK
        ANI     KBDA            ;WAS INT FROM KEYBOARD?
        CNZ     IIKB            ;YES, HANDLE IT
        POP     PSW             ;RESTORE STATUS
        EI                      ;RE-ENABLE INTERRUPTS
        RET                     ;RETURN
;
;  THE FOLLOWING HANDLES A KEYBOARD INTERRUPT.
;
IIKB:   IN      KB              ;GET THE KEYBOARD DATA
        XRI     177             ;TAKE ONES COMPLEMENT
        CMI     4               ;IS IT CONTROL D?
        JNZ     IIKBGC          ;NO, SEND IT
        MVI     A,-1            ;GET A -1
```

```
        STA     FKILL           ;SET KILL FLAG
        RET
IIKBGC: OUT     MODEM           ;TRANSMIT IT
        RET                     ;RETURN
;
;  THE FOLLOWING HANDLES A MODEM INPUT INTERRUPT
;
IIRDA:  IN      MODEM           ;GET INPUT DATA
        ANI     177             ;REMOVE PARITY
        PUSH    H               ;SAVE HL
        PUSH    D               ;SAVE DE
        CMI     101              ;IS IT A CAPITAL LETTER?
        JC      IINLET           ;NO, LEAVE ALONE
        CMI     133             ;IF GREATER THAN Z, NOT A LETTER.
        JNC     IINLET          ;NC, NOT A LETTER
        ORI     40              ;MAKE SURE IT IS LOWER CASE
IINLET: PUSH    PSW             ;SAVE THE CHAR
        LHLD    BUFIN           ;GET INPUT QUEUE INDEX
        XCHG                    ;PUT IT IN DE
        LHLD    BUFOUT          ;GET OUT INDEX
        MOV     A,D             ;GET LEFT OF IN INDEX
        SUB     H               ;ARE THEY EQUAL?
        MOV     H,A             ;SAVE INTERMEDIATE RESULT
        MOV     A,E             ;GET RIGHT OF IN INDEX
        SUB     L               ;IF EQUAL, Q IS FULL
        ORA     H               ;COMBINE INTERMEDIATE RESULT
        JZ      IIERR           ;EQUAL?  IF SO, ERROR
;
        MVI     A,3CC           ;GET TONE OFF  CODE
        OUT     CSC             ;TURN OFF ERROR TONE
;
        LXI     H,BUF           ;GET BUFFER START ADDR
        DAD     D               ;ADD INDEX TO BUFFER START
        POP     PSW             ;GET DATA CHAR BACK
        MOV     M,A             ;PUT DATA IN Q
        XCHG                    ;PUT INDEX BACK IN HL
        INX     H               ;ADD 1
        MOV     A,H             ;GET LEFT HALF
        ANI     BPAGES          ;TAKE MOD BUFFER SIZE
        MOV     H,A             ;PUT BACK IN HL
        SHLD    BUFIN           ;PUT INDEX BACK
;
        POP     D               ;RESTORE DE
        POP     H               ;RESTORE HL
        RET                     ;RETURN
;
IIERR:  MVI     A,301           ;GET TONE ON CODE
        OUT     CSC             ;TURN ON ERR TONE
        POP     PSW             ;REMOVE DATA
        POP     D               ;RESTORE DE
        POP     H               ;RESTORE HL
```

```
        RET                             ;RETURN
;
        .SBTTL  FLAGS AND GLOBAL DECLARATIONS
;
BUF     =       12000                   ;ADDRESS OF START OF INPUT QUEUE
BPAGES  =       3                       ;NUMBER OF 256 BYTE PAGES -1
BUFIN:  .BYTE   1,0                     ;CHARACTER QUEUE INPUT INDEX
BUFOUT: .BYTE   0,0                     ;REMOVAL INDEX FOR THE QUEUE
WTIME   =       1                       ;NUMBER OF SECONDS TO WAIT FOR
INPUT
BKILL:  .BYTE   0                       ;KILL THE LINE FLAG
SPAUSE  =       4                       ;SHORT PAUSE PHONEME
LPAUSE  =       77                      ;LONG PAUSE PHONEME
AEIOUY: .ASCIZ  /aeiouy/
WORDS   =       33764                   ;ADDR OF START OF PDINTER TABLE
TTOP    =       WORDS                   ;ADDR OF NUMBER OF WORDS IN TABLE
CFKPT:  .ASCIZ  /cfkpt/
SPHON   =       40                      ;PHONEME FOR S
ZPHON   =       23                      ;PHONEME FOR Z
AEIO:   .ASCIZ  /aeio/
AEO:    .ASCIZ  /aeo/
CG:     .ASCIZ  /cg/
AEIOU:  .ASCIZ  /aeiou/
AEIOUQ: .ASCIZ  /aeiouy?/
BCDFGKP: .ASCIZ /bdfgkpt/
BCDFGM: .ASCIZ  /bcdfgmnprst/
AEINOR: .ASCIZ  /aeincruy?/
AEHIOU: .ASCIZ  /aehiouwxy/
AIOUYU: .ASCIZ  /aiouyU/
AEIU:   .ASCIZ  /aeiu/
AEIMOU: .ASCIZ  /aeimcuy/
IY:     .ASCIZ  /iy/
IE:     .ASCIZ  /ie/
AOU:    .ASCIZ  /aou/
AEIWXY: .ASCIZ  /aeiouwxy/
BCDFGK: .ASCIZ  /bcdfgkpt/
AEIXYQ: .ASCIZ  /aeiouwxy?/
BDFGKP: .ASCIZ  /bdfgkpt/
IUY:    .ASCIZ  /iuy/
;
        .SBTTL  SPEAK, DRIVING ROUTINE
;
;  THE FOLLOWING ROUTINE DIRECTS THE TRANSLATION FROM ENGLISH
;  TEXT TO PHONEMES FOR USE BY THE VOTRAX.  THE PHONEMES ARE
;  THEN SENT TO THE VOTRAX FOR OUTPUT.
;  THIS ROUTINE IS THE MAIN ROUTINE OF THE WHOLE PROGRAM.
;
SPWORK: .BLKB   200.                    ;WORK BUFFER
        .BYTE   LPAUSE                  ;LONG PAUSE PHON
SIOUT:  .BLKB   50.                     ;OUTPUT BUFFER FOR PHONS
;
```

```
SPEAK:  LXI     SP,70           ;SET STACK POINTER
        MVI     A,B300          ;GET 300 BAUD CODE
        OUT     STATUS          ;SET SPEED
        XRA     A               ;CLEAR ACC
        STA     70              ;SET INTERRUPT BYTES WITH NO OPS
        STA     71              ;SET MORE OF INT ADDR
        STA     72              ;SET REST OF IT
        MVI     A,VOUT          ;GET VOTRAX OUTPUT COMMAND
        OUT     VCNTRL          ;INITIALIZE VOTRAX
        EI                      ;ENABLE INTERRUPTS
;  FIRST, MOVE CHARS INTO WORK BUFFER, REMOVING BLANKS AND
;  TABS.  ALSO, PLACE A MARK OF ZERO AFTER EACH WORD OR SPECIAL
;  SYMBOL.  PLACE DOUBLE MARK AT END OF STRING.
SPOVER: LXI     H,SPWORK        ;GET ADDR OF WORK BUF
        XRA     A               ;GET A ZERO
        STA     FKILL           ;CLEAR KILL FLAG
SPLP1:  CALL    GET             ;GET A CHAR
SPLP2:  ORA     A               ;IS IT THE TERMINATOR
        JZ      SPNEXT          ;IF YES, GO WITH THIS LINE
        CMI     41              ;IS IT A CONTROL CHAR?
        JC      SPLP1           ;IF YES, IGNORE IT
SPLP3:  MOV     M,A             ;PUT CHAR IN WORK
        INX     H               ;INC POINTER
        CMI     141             ;IS IT LESS THAN A?
        JC      SPMRK2          ;YES, TERMINATE WORD
        CMI     173             ;LESS THAN Z?
        JNC     SPMRK2          ;NC, INSERT THE MARK
SPLP3E: CALL    GET             ;GET THE NEXT CHAR
        CMI     141             ;IS IT A LETTER TOO?
        JC      SPMARK          ;NC, INSERT A MARK
        CMI     173             ;IS IT A LETTER?
        JNC     SPMARK          ;NC, INSERT A MARK
        MOV     M,A             ;INSERT IN BUFFER
        INX     H               ;INC BUFFER POINTER
        JMP     SPLP3E          ;TRY NEXT CHAR
SPMRK2: CALL    GET             ;GET NEXT CHAR
SPMARK: MVI     M,0             ;INSERT THE MARK
        INX     H               ;INC POINTER
        JMP     SPLP2           ;FIND ANOTHER WORD
SPNEXT: MVI     M,0             ;INSERT A MARK
        INX     H               ;INC POINTER
        MVI     M,0             ;INSERT SECOND MARK
;
;  NOW WORK WITH EACH WORD UNTIL DONE OR UNTIL THE KILL KEY IS PRESSED
;
        LXI     D,SPWORK        ;GET ADDR OF WORD BUF
SPLP4:  LXI     B,SPOUT         ;GET OUTPUT BUF ADDR
        LDAX    D               ;GET THE NEXT CHAR
        ORA     A               ;IS IT A MARK?
        JZ      SPOVER          ;IF YES, QUIT AND START OVER
        IN      377             ;IS SPELL ONLY FLAG SET?
```

```
          ORA     A                 ;SET CONDITION CODE
          JM      SPSPL             ;YES, GO DIRECTLY TO SPELL
          CALL    PHREAD            ;SEE IF ENTIRE WORD IS IN TABLE
          CM      PHPRON            ;IF NOT, TRY PRONOUNCIATION
SPSPL:    CM      PHSPELL           ;IF NOT, SPELL IT
          MVI     A,SPAUSE          ;GET SHORT PAUSE PHON
          STAX    B                 ;PUT IN OUTPUT
          INX     B                 ;BUMP POINTER
          XRA     A                 ;GET A MARK
          STAX    B                 ;PUT IN OUTPUT
          LXI     B,SPOUT-1         ;GET ADDR OF OUTPUT
          CALL    VOTRAX            ;SEND TO VOTRAX
;
; MOVE WORK POINTER PAST USED WORD
;
SPLP5:    INX     D                 ;INC POINTER
          LDAX    D                 ;GET NEXT CHAR
          ORA     A                 ;IS IT THE MARK?
          JNZ     SPLP5             ;NO, KEEP LOOKING
          INX     D                 ;POINT TO START OF NEXT WORD
          LDA     PKILL             ;HAS KILL BEEN ENTERED?
          ORA     A                 ;SET CONDITION CODE
          JP      SPLP4             ;NO, KEEP GOING
          JMP     SPOVER            ;GO START OVER AGAIN
;
          .SBTTL  GET, INPUT CHARACTER DEQUEUEING ROUTINE
;
;  THIS ROUTINE IS CALLED BY SPEAK WHEN AN INPUT CHATACTER
;  FROM THE MODEM IS NEEDED.
;  IF A CHARACTER IS IN THE QUEUE, IT IS RETURNED.
;  IF THE CHARACTER IS A LINE FEED OR RETURN, IT IS SENT AS A LINE
TERMINATION MARK.
;  IF THE QUEUE IS EMPTY, A TIMER IS STARTED.  IF NO CHARACTER IS
;  FORTHCOMING BEFORE THE TIMER RUNS OUT, THE END OF THE LINE
;  IS ASSUMED AND A TERMINATOR IS RETURNED.
;
GET:      PUSH    B                 ;SAVE BC
          PUSH    D                 ;SAVE DE
          PUSH    H                 ;SAVE HL
;
          MVI     B,WTIME           ;GET PRIMARY TIMER
GELP1:    LXI     H,77777           ;GET SECONDARY TIMER
GELP2:    PUSH    H                 ;SAVE TIMER
          LHLD    BUFOUT            ;GET BUFFER Q REMOVAL INDEX
          INX     H                 ;ADD 1
          MOV     A,H               ;GET HIGH PART
          ANI     BPAGES            ;GET MOD BUFFER Q SIZE
          MOV     D,A               ;PUT IN DE
          MOV     E,L               ;PUT REST IN DE
          LHLD    BUFIN             ;GET INPUT INDEX
          SUB     H                 ;IF EQUAL, Q IS EMPTY
```

```
        MOV     H,A             ;SAVE INTERMEDIATE RESULT
        MOV     A,E             ;GET RIGHT PART OF INDEX
        SUB     L               ;CMP WITH RIGHT OF IN INDEX
        ORA     H               ;COMBINE RESULT
        JNZ     GEYES           ;YES, THERE IS A CHAR IN THE Q
;
        POP     H               ;RESTORE TIMER
        DCX     H               ;DEC COUNTER
        MOV     A,H             ;GET HALF OF TIMER
        ORA     L               ;IS TIMER ZERO?
        JNZ     GELP2           ;NO, WAIT SOME MORE
;
        DCR     B               ;DEC PRIMARY TIMER
        JNZ     GELP1           ;IF NOT ZERO, WAIT MORE
;
GEMARK: XRA     A               ;GET A TERMINATOR MARK
        POP     H               ;RESTORE HL
        POP     D               ;RESTORE DE
        POP     B               ;RESTORE BC
        RET                     ;RETURN
;
GEYES:  LXI     H,BUF           ;GET BUFFER ADDR
        DAD     D               ;ADD INDEX
        MOV     A,M             ;GET THE CHAR
        XCHG                    ;GET INDEX IN HL
        SHLD    BUFOUT          ;PUT INDEX BACK
;
        CMI     12              ;IS THIS CHAR A LINE FEED?
        POP     H               ;REMOVE TIMER FROM STACK
        JZ      GEMARK          ;IF A LINE FEED, RETURN A MARK
        CMI     15              ;IS IT A CARRIAGE RETURN?
        JZ      GEMARK          ;IF SO, ALSO RETURN A MARK
;
        POP     H               ;RESTORE HL
        POP     D               ;RESTORE DE
        POP     B               ;RESTORE BC
        RET                     ;RETURN
;
        .SBTTL  VOTRAX I/O DRIVER
;
;  THIS ROUTINE WAITS UNTIL THE  VOTRAX UNIT IS NOT BUSY.
;  THEN THE PHONEMES IN THE STRING WHOSE ADDRESS IS IN
;  BC ARE PASSED TO THE VOTRAX AND OUTPUT IS BEGUN.
;
;  NOTE THAT ONE IS SUBTRACTED FROM EACH PHONEME IN THE STRING
;  BEFORE IT IS PASSED TO THE VOTRAX.  THIS IS BECAUSE THE
;  PHONEMES ARE STORED ONE GREATER THAN THEIR ACTUAL CODE
;  SO THAT A ZERO MAY BE USED AS THE TERMINATION MARK.
;
VOTRAX: PUSH    B               ;SAVE BC
VWAIT:  IN      STATUS          ;GET STATUS BITS
```

```
        ANI     VBUSY           ;IS VOTRAX BUSY?
        JNZ     VWAIT           ;YES, WAIT
;
        MVI     A,VIN           ;GET INPUT CODE
        OUT     VCNTRL          ;SEND TO VOTRAX
;
VLOOP:  LDAX    B               ;GET A PHONEME
        ORA     A               ;IS THIS THE MARK?
        JZ      VDONE           ;YES, EXIT
        DCR     A               ;GET CORRECT PHONEME
        OUT     VDATA           ;SEND TO VOTRAX
        INX     B               ;GET NEXT PHONEME ADDR
        JMP     VLOOP           ;KEEP GOING UNTIL DONE
;
VDONE:  MVI     A,VOUT          ;GET OUTPUT CODE
        OUT     VCNTRL          ;START IT UP
;
        POP     B               ;RESTORE BC
        RET                     ;RETURN
;
        .SBTTL  PHREAD, FIND ENTIRE WORD IN PHONEME TABLE
;
;  THE FOLLOWING ROUTINE CHECKS TO SEE IF AN ENTIRE WORD IS
;  IN THE EXCEPTION TABLE.  IF SO, THE PHONEMES FOR THE WORD
;  ARE COPIED INTO THE OUTPUT BUFFER.
;  ON ENTRY, DE CONTAINS THE ADDRESS OF THE WORD TO BE LOOKED UP,
;  BC CONTAINS THE ADDRESS OF THE NEXT AVAILABLE BYTE IN THE OUTPUT
;  BUFFER.
;  ON EXIT, BC CONTAINS THE ADDRESS OF THE NEXT BYTE IN
;  THE OUTPUT BUFFER.  IF THE WORD IS FOUND, 0 IS RETURNED
;  IN THE ACC.  IF NOT FOUND, -1 IS RETURNED.
;
PHREAD: PUSH    D               ;SAVE DE
        PUSH    B               ;SAVE BC
        CALL    PREFIX          ;CALL TABLE SEARCH
        JP      PRYES           ;ACC IS + IF ENTIRE WORD IS FOUND
        POP     B               ;RESTORE BC
        POP     D               ;RESTORE DE
        RET                     ;RETURN WITH -1 IN ACC
;
;
;  WORD HAS BEEN FOUND IN TABLE.  SKIP PAST WORD TO FIND
;  PHONEMES.
;
PRYES:  POP     D               ;PUT OUT BUFFER ADDR IN DE
        MOV     H,B             ;GET HIGH PART OF ADDR
        MOV     L,C             ;GET REST OF TABLE ADDR
PRSKIP: MOV     A,M             ;GET A BYTE
        INX     H               ;BUMP ADDR
        ORA     A               ;IS IT THE END OF THE WORD?
        JNZ     PRSKIP          ;NO, SKIP ALL OF WORD
```

```
;
;  HL NOW POINT TO FIRST BYTE OF PHONEMES
;
        CALL    MOVE                ;MOVE PHONS INTO OUTPUT BUF
        MOV     B,D                 ;MOVE OUTPUT BUFFER ADDR BACK TO BC
        MOV     C,E                 ;MOVE REST OF IT
        POP     D                   ;RESTORE ADDR OF WORD
        XRA     A                   ;SET SUCCESS FLAG
        RET                         ;RETURN
;
        .SBTTL  SEARCH FOR PREFIX
;
;  THIS ROUTINE SEARCHES FOR THE INPUT WORD OR THE LONGEST PREFIX
;  OF THE INPUT WORD.  ON ENTRY, DE CONTAINS THE ADDRESS OF
;  THE WORD TO BE LOOKED UP.  ON EXIT, BC CONTAINS THE ADDRESS WITHIN
;  THE STRING TABLE WHERE THE WORD OR PREFIX WAS FOUND.
;  IF THE ENTIRE WORD IS FOUND, 0  IS RETURNED IN THE ACC.
;  IF THE ENTIRE WORD IS NOT FOUND, -1 IS RETURNED IN THE ACC.
;  IF NO MATCH ON EITHER WORD OR PREFIX IS FOUND, BC RETURNS 0.
;
PXBOT:  .BLKB   2                   ;ONE BELOW CURRENT TABLE
PXTOP:  .BLKB   2                   ;ONE ABOVE CURRENT TABLE TOP
PXI:    .BLKB   2                   ;THE I TH TABLE ELEMENT
PXPREF: .BLKB   2                   ;THE ADDR OF THE LONGEST PREFIX SO
FAR
PXIN:   .BLKB   2                   ;THE ADDRESS OF THE INPUT WORD
PXADDR: .BLKB   2                   ;THE ADDR OF THE ITH TABLE ELEMENT
;
PREFIX: PUSH    D                   ;SAVE DE
        XCHG                        ;MOVE INPUT ADDR TO HL
        SHLD    PXIN                ;SAVE IN PXIN
        MVI     H,0                 ;CLEAR H
        MOV     L,H                 ;CLEAR L
        SHLD    PXBOT               ;CLEAR BOT
        SHLD    PXPREF              ;SET LAST PREFIX TO 0
        MVI     B,1                 ;SET MINIMUM LENGTH OF PREFIX
;
;  B WILL CONTAIN THE MINIMUM NUMBER OF CHARACTERS NECESSARY TO
;  SELECT A NEW PREFIX.  I.E., INITIALLY, A PREFIX MUST BE
;  AT LEAST TWO CHARACTERS.
;
        LHLD    TTOP                ;GET TABLE TOP
        SHLD    PXTOP               ;SET PXTOP TO TABLE TOP
;
;  FIRST, CALCULATE I=(BOT+TOP)/2
;
PXLP1:  LHLD    PXBOT               ;GET TABLE BOTTOM
        XCHG                        ;PUT IT IN DE
        LHLD    PXTOP               ;GET TABLE TOP
        DAD     D                   ;ADD BOT TO TOP
        MOV     A,H                 ;GET LEFT HALF OF RESULT
```

```
        RAR                         ;SHIFT IT RIGHT
        MOV     H,A                 ;PUT RESULT BACK
        MOV     A,L                 ;GET RIGHT HALF
        RAR                         ;SHIFT IT RIGHT
        MOV     L,A                 ;PUT IT BACK
        SHLD    PXI                 ;STORE RESULT IN I
;
;   IF BOT < I THEN CONTINUE
;
        MOV     A,E                 ;GET RIGHT HALF OF BOT
        SUB     L                   ;SUB RIGHT HALF OF I
        MOV     A,D                 ;GET LEFT HALF OF BOT
        SBB     H                   ;IS BOT<I?
        JC      PXLP2               ;YES, CONTINUE
;
;   IF NOT, THE TABLE HAS BEEN EXHAUSTED AND THE LONGEST PREFIX SO FAR
;   SHOULD BE RETURNED (PXPREF).
;
        LHLD    PXPREF              ;GET ADDR OF PREFIX
        DRI     -1                  ;SET FLAG TO INDICATE PREFIX
PXEXIT: MOV     B,H                 ;GET RETURN ADDR IN BC
        MOV     C,L                 ;MOVE REST
        POP     D                   ;RESTORE DE
        RET                         ;RETURN FROM ROUTINE
;
;   NOW CALCULATE ADDR OF ITH TABLE ELEMENT
;
PXLP2:  XCHG                        ;MOVE I TO DE
        LXI     H,WORDS             ;GET ADDR OF START OF TABLE
        DAD     D                   ;ADD I
        DAD     D                   ;ADD I (ADDR=TABLE+2I)
        MOV     E,M                 ;GET FIRST BYTE FROM TABLE
        INX     H                   ;GET ADDR NEXT BYTE
        MOV     D,M                 ;GET SECOND BYTE
        XCHG                        ;PUT TABLE VALUE IN HL
        SHLD    PXADDR              ;SAVE ITH ELEMENT ADDR
        XCHG                        ;MOVE I TO DE
        LHLD    PXIN                ;GET ADDR OF INPUT WORD
        MVI     C,0                 ;CLEAR COUNTER OF MATCHED CHARS
;
;   THE FOLLOWING LOOP WILL BE EXECUTED AS LONG AS CONSEQUTIVE
;   CHARACTERS OF THE INPUT WORD MATCH THOSE OF THE TABLE ELEMENT,
;   OR UNTIL ONE OR THE OTHER OF THESE STRINGS ARE TERMINATED.
;   THE NUMBER OF MATCHED CHARACTERS WILL BE COUNTED IN C.
;
PXLP3:  LDAX    D                   ;GET A TABLE CHAR
        CMP     M                   ;COMPARE WITH INPUT CHAR
        JZ      PXEQU               ;IF THEY ARE EQUAL
        JC      PXLT                ;IF TAB CHAR < INPUT CHAR
;
;   IF THE TABLE ELEMENT CHARACTER IS GREATER THAN THE INPUT CHARACTER
```

```
;  THE TOP HALF OF THE TABLE SHOULD BE CHOPPED OFF AND THE
;  SUB-TABLE SEARCHED.
;
        LHLD    PXI             ;GET I
        SHLD    PXTOP           ;TOP=I  CHOPS TOP HALF
        JMP     PXLP1           ;SEARCH THE SUB-TABLE
;
;  THE CHARACTERS ARE EQUAL.  SEE IF THE TABLE STRING HAS BEEN
;  EXHAUSTED.
;
PXEQU:  INX     D               ;GET ADDR OF NEXT TAB CHAR
        INX     H               ;GET ADDR OF NEXT INPUT CHAR
        INR     C               ;ADD 1 TO LENGTH COUNT
        LDAX    D               ;GET A TABLE CHAR
        ORA     A               ;IS IT A MARK?
        JNZ     PXLP3           ;NO, KEEP CHECKING CHARS
;
;  IF THE INPUT STRING IS EXHAUSTED TOO, THE ENTIRE WORD HAS
;  BEEN FOUND.  GET ITS ADDRESS AND RETURN WITH FLAG INDICATING
;  ENTIRE WORD WAS FOUND.
;
        CMP     M               ;IS NEXT INPUT CHAR A MARK?
        JNZ     PXEND           ;NO, A PREFIX HAS BEEN FOUND
        LHLD    PXADDR          ;GET ADDR OF TABLE ELEMENT
        XRA     A               ;SET WHOLE WORD FLAG
        JMP     PXEXIT          ;RETURN
;
;  A PREFIX HAS BEEN FOUND.  REMEMBER WHERE IT WAS AND SET THE
;  NECESSARY LENGTH OF A LONGER PREFIX TO ONE GREATER THAN
;  THE LENGTH OF THIS PREFIX.  ALSO, THROW AWAY ALL OF THE
;  TABLE BELOW THIS ELEMENT.
;
PXEND:  LHLD    PXI             ;GET TABLE INDEX
        SHLD    PXBOT           ;CHOP BY BOT=I
        LHLD    PXADDR          ;GET ADDR OF PREFIX
        SHLD    PXPREF          ;REMEMBER IT
        MOV     B,C             ;SET NEW NECESSARY LENGTH
        INR     B               ;TO THIS LEN + 1
        JMP     PXLP1           ;SEARCH FOR LONGER PREFIX
;
;  THE FOLLOWING IS REACHED IF THE TABLE ELEMENT CHARACTER IS LESS
;  THAN THE INPUT CHARACTER.  IF THE LENGTH OF THE CURRENT
;  MATCH IS LESS THAN OR EQUAL TO THE NECESSARY LENGTH OF A
;  NEW PREFIX, THROW AWAY THE BOTTOM HALF OF THE TABLE AND
;  SEARCH THE SUB-TABLE.
;
PXLT:   MOV     A,B             ;GET NEC LENGTH FOR NEW PREFIX
        CMP     C               ;COMPARE WITH CURRENT LEN
        JNC     PXYES           ;IF <= GO CHOP TABLE
;
;   IF THE CURRENT MATCH LENGTH IS GREATER THAN THE NECESSARY
```

```
;    LENGTH FOR A LONGER PREFIX, CACULATE A NEW 'I' BY
;   I=(BOT+I)/2, SO THAT OLD BOT IS NOT LOST IN CASE
;   OF FAILURE TO MATCH SUBSEQUENT CHARACTERS OF THE INPUT
;   STRING.
;
        LHLD    PXBOT           ;GET BOT
        XCHG                    ;PUT IN DE
        LHLD    PXI             ;GET I
        DAD     D               ;ADD BOT TO I
        MOV     A,H             ;GET HIGH BYTE OF RESULT
        RAR                     ;SHIFT RIGHT
        MOV     H,A             ;PUT IT BACK
        MOV     A,L             ;GET LOW BYTE OF RESULT
        RAR                     ;SHIFT IT RIGHT
        MOV     L,A             ;PUT IT BACK
        SHLD    PXI             ;PUT RESULT IN I
;
;   IF NEW I IS GREATER THAN BOT, THEN SEARCH THIS SUB-TABLE.
;
        MOV     A,E             ;GET RIGHT PART OF BOT
        SUB     L               ;SUBTRACT RIGHT OF I
        MOV     A,D             ;GET LEFT OF BOT
        SBB     H               ;IS I>BOT?
        JC      PXLP2           ;YES, SEARCH SUB-TABLE
;
;   IF NOT, THE SUB-TABLE IS EXHAUSTED.  THERE ARE NO PREFIXES
;   OF THE CURRENT NECESSARY LENGTH.  THEREFORE, INCREMENT
;    NECESSARY LENGTH, BUMP TABLE BOTTOM PAST CURRENT ELEMENT,
;   AND SEARCH THE EXPANDED SUB-TABLE.
;
        LHLD    PXBOT           ;GET OLD BOTTOM
        INX     H               ;INC BOT
        SHLD    PXBOT           ;PUT IT BACK
        INR     E               ;INC NEC LENGTH
        JMP     PXLP1           ;SEARCH EXPANDED TABLE
;
;   IF NUMBER OF CHARACTERS MATCHED WAS LESS THAN THE NECESSARY
;   LENGTH OF A LONGER PREFIX, CHOP OFF BOTTOM HALF OF TABLE
;   AND CONTINUE SEARCH.
;
PXYES:  LHLD    PXI             ;GET I
        SHLD    PXBOT           ;SET BOT=I TO CHOP
        JMP     PXLP1           ;SEARCH SUB-TABLE
;
        .SBTTL  PHPRON,  DIRECTS AUTOMATIC PRONUNCIATION
;
;   THIS ROUTINE DIRECTS THE ATTEMPT AT AUTOMATIC PRONUNCIATION.
;   ON ENTRY , BC CONTAINS THE ADDRESS OF THE
;   AVAILABLE BYTE IN THE OUTPUT BUFFER,  AND DE CONTAINS THE
;   ADDRESS OF THE WORD TO BE PRONOUNCED.  ON EXIT, BC RETURNS
;   THE ADDRESS OF THE NEXT AVAILABLE BYTE IN THE OUTPUT BUFFER.
```

```
;  IF SUCCESSFUL, THE ACC RETURNS WITH ZERO.  IF AUTOMATIC
PRONOUNCIATION
;  FAILS, -1 IS RETURNED IN THE ACC.
;
PPSFLG:  .BLKB   1                ;SAVE FINAL S RETURN CODE
PPHCLD:  .BLKB   60.              ;WORK BUFFER
;
PHPRON:  PUSH    D                ;SAVE ADDR INPUT WORD
         PUSH    B                ;SAVE BUFFER ADDR
         PUSH    E                ;SAVE A WORK COPY
         LXI     H,PPHCLD+6       ;GET ADDR WORK BUFFER
         XCHG                     ;SET UP PARMS FOR MOVE
         CALL MOVE                ;MOVE WORD TO WORK BUFFER
         LXI     E,PPHCLD+6       ;PUT WORD START ADDR IN BC
;
;  THE INPUT WORD HAS BEEN MOVED TO THE WORK BUFFER.  BC POINTS
;  TO THE FIRST CHAR IN THE WORD, DE POINTS TO THE TERMINATING
;  MARK AT THE END OF THE WORD.  IF WORD IS ONLY ONE LETTER OR
;  CONTAINS NO VOWELS, IT CANNOT BE PRONOUNCED.  EXIT WITH FAILURE.
;
         CALL    VOWEL            ;ARE THERE ANY VOWELS?
         JM      PPFAIL           ;NO, FAIL
         DCX     D                ;GET ADDR OF LAST CHAR
         MOV     A,E              ;GET RIGHT PART OF END ADDR
         SUB     C                ;CMP WITH RIGHT PART OF START
         PUSH    H                ;SAVE HL
         MOV     H,A              ;SAVE INTERMEDIATE RESULT
         MOV     A,D              ;GET LEFT OF END ADDR
         SBB     B                ;IS THERE ONY ONE LETTER?
         ORA     H                ;COMBINE WITH REST
         POP     H                ;RESTORE HL
         JZ      PPFAIL           ;IF YES, FAIL
;
;  CHECK FOR AND REMOVE FINAL S.
;
         CALL    FINALS           ;PROCESS FINAL S
         STA     PPSFLG           ;SAVE RETURN CODE
         ORA     A                ;WAS THERE A FINAL S?
         JZ      PPNOS            ;NO
;
;  SINCE CHANGES WERE MADE, TRY DIRECT LOOK UP AGAIN.
;
         XCHG                     ;PUT END ADDR IN HL
         MOV     E,C              ;MOVE START ADDR TO DE
         MOV     D,B              ;MOVE REST
         POP     B                ;GET BUFFER ADDR BACK
         PUSH    H                ;SAVE END ADDR IN CASE FAILURE
         CALL    PHREAD           ;SEE IF IT IS IN TABLE
         JM      PPTRYA           ;IF NOT, TRY AUTO PRON
;
;  IF FOUND, ADD THE PHONEME FOR THE STRIPPED S.
```

```
;
        LDA     PPSFLG          ;GET FINAL S RETURN CODE
        STAX    B               ;INSERT PHONEME IN OUTPUT
        INX     B               ;INC BUFFER POINTER
        MVI     A,SPHCN         ;GET PHONEME FOR S
        STAX    B               ;PUT IN OUTPUT BUFF
        INX     B               ;INC BUFFER END
        JMP     PPDONE          ;RETURN SUCCESSFULLY
;
;  AFTER FINAL S CHANGES, THE WORD WAS STILL NOT FOUND.
;  RESTORE POINTERS AND CONTINUE IN ATTEMPT TO AUTO PRONOUNCE.
;
PPTRYA: POP     H               ;RESTORE END POINTER
        PUSH    B               ;RESAVE OUTPUT BUF ADDR
        MOV     B,D             ;START BACK IN BC
        MOV     C,E             ;MOVE REST
        XCHG                    ;PUT END ADDR BACK IN DE
;
;  CALL ALL PREPROCESSING ROUTINES.
;
PPNOS:  DCX     B                ;GET ADDR CHAR BEFORE WORD
        XRA     A                ;GET A ZERO
        STAX    B               ;PLACE IN FRONT OF WORD
        INX     B               ;RESTORE POINTER
        CALL    MIDU            ;MIDDLE U PROCESSOR
        CALL    FINALE          ;FINAL E PROCESSOR
        CALL    MIDE            ;MIDDLE E PROCESSOR
        CALL    MIDS            ;CALL MIDDLE S PROCESSOR
;
;  IF S WAS STRIPPED, REPLACE IT
;
        LDA     PPSFLG          ;GET FINAL S FLAG
        ORA     A               ;WAS THERE A FINAL S?
        JZ      PPNREP          ;NO, DO NOT REPLACE IT
        INX     D               ;ADJUST WORD END
        MVI     A,'s            ;GET AN S
        STAX    D               ;PUT ON END OF WORD
;
;  NOW PLACE BEGINNING AND END OF WORD MARKERS ON WORD
;
PPNREP: DCX     B               ;EXTEND WORD START
        MVI     A,'#            ;GET THE MARKER
        STAX    B               ;INSERT IN FRONT OF WORD
        INX     D               ;EXTEND WORD END
        STAX    D               ;MARK END OF WORD TOO
        INX     D               ;EXTEND WORD END
        XRA     A               ;GET A TERMINATION MARK
        STAX    D               ;MARK END OF WORD
;
;  NOW THE LONGEST PREFIX OF THE WORD IS FOUND.  THE
;  PREFIX IS REMOVED, THE PHONEME EQUIVALENT FOR THE
```

```
;  PREFIX IS COPIED INTO THE OUTPUT BUFFER, AND
;  ANY REPLACEMENT CHARACTERS FOUND IN THE TABLE ARE PLACED BACK ON
;  THE BEGINNING OF THE WORD.
;  THE PROCESS IS CONTINUED UNTIL THE INPUT WORD IS EXHAUSTED,
;  IN WHICH CASE AUTO PRONOUNCIATION IS SUCCESSFUL, OR UNTIL A PREFIX
;  CAN NO LONGER BE FOUND, IN WHICH CASE AUTO PRONOUNCIATION FAILS.
;
        MOV     D,B             ;PUT START OF WORD IN DE
        MOV     E,C             ;MOVE REST
PPLP1:  LDAX    D               ;GET THE NEXT CHAR
        ORA     A               ;IS WORD EXHAUSTED
        JZ      PPDONE          ;YES, SUCCESS
;
;  PLACE WORD FRAGMENT MARKER IN FRONT OF WORD.
;
        DCX     D               ;EXTEND FRONT OF WORD
        MVI     A,'%            ;GET THE MARKER
        STAX    D               ;PLACE ON WORD
        CALL    PREFIX          ;FIND LONGEST PREFIX
        MOV     A,B             ;GET FIRST HALF OF PREFIX ADDR
        ADD     C               ;IF BC=0, NO PREFIX
        JZ      PPFAIL          ;IF SO, FAILURE
;
;  REMOVE PREFIX FROM WORD BY INCREMENTING INDEX POINTER
;
PPLP2:  LDAX    B               ;GET A PREFIX CHAR
        ORA     A               ;IS THIS END OF PREFIX?
        JZ      PPL2EX          ;YES, DONE
        INX     D               ;SKIP CHAR IN WORD
        INX     B               ;SKIP CHAR IN PREFIX
        JMP     PPLP2           ;KEEP SKIPPING CHARS
;
;  POINT BC TO FIRST PHONEME CHARACTER.
;  GET THE OUTPUT BUFFER ADDRESS IN HL.
;
PPL2EX: INX     B               ;POINT BC TO FIRST PHON
        POP     H               ;GET OUTPUT BUF ADDR
;
;  COPY PHONEMES TO OUTPUT BUFFER UNTIL A TERMINATION MARK IS FOUND,
;  INDICATING END OF PHONEMES, OR SEPARATION MARK (200 OCTAL),
;  INDICATING THAT REPLACEMENT CHARS FOLLOW.
;
PPLP3:  LDAX    B               ;GET A TABLE CHAR
        MOV     M,A             ;PUT IN OUTPUT BUF
        INX     B               ;POINT TO NEXT TAB CHAR
        ORA     A               ;IS IT A TERMINATOR?
        JZ      PPL3EX          ;YES, DONE
        CMI     200             ;IS IT THE SEPARATOR?
        JZ      PPSEP           ;YES, INSERT REP CHARS
        INX     H               ;INC OUTPUT POINTER
        JMP     PPLP3           ;MOVE ANOTHER PHON
```

```
;
;  NOW COPY THE REPLACEMENT CHARACTERS ONTO THE FRONT OF THE WORD.
;
PPSEP:   LDAX    B                  ;GET A REP CHAR
         ORA     A                  ;IS IT THE END MARKER?
         JZ      PPL3EX             ;YES, DONE
         DCX     D                  ;EXTEND WORD FRONT
         STAX    D                  ;INSERT THE REP CHAR
         INX     B                  ;GET ADDR NEXT REP CHAR
         JMP     PPSEP              ;MOVE MORE
;
;  RESAVE OUTPUT BUFFER ADDR
;
PPL3EX:  PUSH    H                  ;SAVE WORK COPY OF ADDR
         JMP     PPLP1              ;LOOK FOR ANOTHER PREFIX
;
PPFAIL:  POP     B                  ;REMOVE WORK BUF ADDR
         POP     B                  ;RESTORE OUTPUT BUF ADDR
         POP     D                  ;RESTORE INPUT ADDR
         ORI     -1                 ;SET FAILURE FLAG
         RET                        ;RETURN
;
PPDONE:  POP     B                  ;RESTORE OUTPUT BUF ADDR
         POP     H                  ;REMOVE ORIGINAL ADDR
         POP     D                  ;RESTORE INPUT ADDR
         XRA     A                  ;SET SUCCESS FLAG
         RET                        ;RETURN
;
         .SBTTL  PHSPELL, CHANGE EACH LETTER TO PHONEMES
;
;  THE FOLLOWING ROUTINE SPELLS A WORD OR SYMBOL BY LOOKING UP THE
;  PHONEMES FOR EACH OF THE LETTERS WHICH MAKE UP THE WORD.
;  ON ENTRY, BC CONTAINS THE ADDRESS OF THE NEXT AVAILABLE
;  BYTE IN THE OUTPUT BUFFER, DE CONTAINS THE ADDRESS OF THE
;  WORD TO BE SPELLED.
;  ON EXIT, BC RETURNS THE ADDRESS OF THE NEXT AVAILABLE BYTE IN
;  BYTE IN THE OUTPUT BUFFER.
;
PSSTR:   .ASCIZ  /* /                ;STRING TO  HOLD LETTERS
;
PHSPELL: PUSH    D                  ;SAVE DE
         PUSH    D                  ;SAVE A WORK COPY
         LXI     D,PSSTR            ;GET ADDR OF STRING TO BE FOUND
PSLOOP:  POP     B                  ;GET ADDR OF NEXT LETTER IN WORD
         MOV     A,M                ;PUT THE CHAR IN ACC
         ORA     A                  ;IS IT THE TERMINATION MARK?
         JZ      PSDONE             ;YES, QUIT
         INX     H                  ;INC POINTER
         PUSH    H                  ;SAVE ADDR FOR NEXT TIME
         LXI     H,PSSTR+1          ;GET ADDR OF CHAR POSITION IN
STRING
```

```
        MOV     E,A             ;PUT CHAR IN TEST STRING
        CALL    PHREAD          ;FIND IT AND MOVE PHONS
        MVI     A,SPAUSE        ;GET SHORT PAUSE PHON
        STAX    B               ;PUT IN OUTPUT BUF
        INX     B               ;BUMP OUTPUT POINTER
        JMP     PSLOOP          ;GO DO NEXT LETTER
;
PSDONE: XRA     A               ;GET A MARK
        STAX    B               ;PUT IN OUTPUT
        POP     D               ;RESTORE DE
        RET                     ;RETURN
;
        .SBTTL  FINAL S PREPROCESSER
;
;   THIS ROUTINE CHECKS THE INPUT WORD TO SEE IF IT ENDS IN AN
;   'S'.  IF A FINAL S IS PRESENT, IT IS REMOVED .
;  REGARDLESS OF THE PRESENCE OF A  FINAL S, A FINAL 'IE' IS
;  CHANGED TO A 'Y'.  ON ENTRY, BC CONTAINS THE ADDRESS OF
;  THE BEGINNING OF THE INPUT WORD AND DE CONTAINS THE ADDRESS
;  OF THE END OF THE WORD.  ON EXIT, DE RETURNS THE ADDRESS
;  OF THE NEW END OF THE WORD AFTER FINAL S  AND IE HAVE BEEN
;  REMOVED.  IF A FINAL  S WAS NOT FOUND, 0 IS RETURNED IN
;  THE ACC, OTHERWISE THE PHONEME FOR A VOICED
;  OR UNVOICED 'S' IS RETURNED.
;
FINALS: PUSH    B               ;SAVE BC
        MVI     B,0             ;SET RETURN FLAG
        LDAX    D               ;LOAD LAST CHAR OF WORD
        CMI     163             ;IS IT AN S?
        JNZ     FSNOS           ;NO, NO FINAL S
;
;  AN S AFTER ANOTHER S OR AFTER A U SHOULD NOT BE REMOVED
;  (E.G. LESS MINUS).
;
        DCX     D               ;MOVE POINTER TO NEXT LEFT CHAR
        LDAX    D               ;LOAD NEXT TO LAST CHAR
        INX     D               ;PUT POINTER BACK
        CMI     163             ;IS IT SS?
        JZ      FSEXIT          ;YES, IGNORE IT AND RETURN
        CMI     165             ;IS IT US?
        JZ      FSEXIT          ;YES, IGNORE AND RETURN
;
;  REMOVE FINAL S NOT PRECEEDED BY S OR U.  SET VOICED S FLAG.
;
        XRA     A               ;GET A MARK
        STAX    D               ;REMOVE FINAL S
        DCX     D               ;RESET END POINTER
        MVI     B,ZPHON         ;SET RETURN  FLAG FOR VOICED S
;
;  IF NEW FINAL LETTER IS ONE OF C F K P OR T, THE FINAL S IS
;   UNVOICED.  IF SO, SET RETURN CODE AND RETURN.
```

```
;
        LDAX    D               ;GET LAST CHAR
        LXI     H,CFKET         ;GET ADDR CF STRING
        CALL    CNEOF           ;IS IT ONE CF THEM?
        JM      FSNOS           ;NC, CHECK FCR IE
        MVI     E,SPHCN         ;YES, SET RETURN CODE
        JMP     FSEXIT          ;RETURN
;
;  NOW CHECK FOR A FINAL IE.    IF FOUND, CHANGE IT TO Y.
;
FSNOS:  LDAX    D               ;GET FINAL LETTER
        CMI     145             ;IS IT AN E?
        JNZ     FSEXIT          ;NC, RETURN
        DCX     D               ;GET ADDR CF NEXT CHAR
        LDAX    D               ;GET THE CHAR
        INX     D               ;PUT POINTER BACK
        SUI     151             ;IS IT IE?
        JNZ     FSEXIT          ;NC, RETURN
;
        STAX    D               ;PUT ZERO IN PLACE CF E
        DCX     D               ;ADJUST END POINTER
        MVI     A,171           ;GET A Y
        STAX    D               ;INSERT ON END OF WORD
;
FSEXIT: MCV     A,B             ;GET RETURN CCDE IN ACC
        ORA     A               ;SET CONDITICN CODE
        PCP     B               ;RESTCRE BC
        RET                     ;RETURN
;
        .SBTTL  FINAL E PREPRCCESSOR
;
;  THIS ROUTINE LCCATES AND MARKS POTENTIAL LCNG VCWELS,
;  PRIMARILY LCNG E'S.  IF AISC LOCATES AND SEPARATES STANDARD
;  SUFFIXES.  CN ENTRY, BC CCNTAINS THE ADDRESS CF THE BEGINNING
;  OF THE WORD TC BE PROCESSED AND DE CONTAINS THE ADDRESS OF
;  THE END CF THE WORD.  NCTE THAT SINCE INSERT MAY BE CALLED,
;  THE END CF THE WORD MIGHT BE EXTENDED TO ALLCW SPACE FOR THE BREAK
;  MARK.
;
;  FIRST, IF THE LAST IETTER CF THE WCRD IS AN E AND IT IS THE ONLY
;  VOWEL IN THE WCRD, IT IS LCNG.  MARK IT AND RETURN.
;
FINALE: LDAX    D               ;GET LAST CHAR
        CMI     145             ;IS IT AN E?
        JNZ     FENC            ;NO, CONTINUE WITH ROUTINE
        CALL    VCWEL           ;ANY CTHER VCWELS?
        JP      FENO            ;YES, CONTINUE
        MVI     A,'E            ;GET A CAP E
        STAX    D               ;PUT IN WORD
        RET                     ;DCNE, SC RETURN
;
```

```
;  NOW CHECK FOR ANY OF THE SUFFIXES IN SUFF0.
;
FEND:   LXI     H,SUFF0         ;GET ADDR OF SUFF0 STRING
        CALL    SUFFIX          ;ANY MATCHES?
        PUSH    D               ;SAVE END ADDR
        XCHG                    ;PUT RETURNED ADDR IN DE
        CP      LONGE           ;IF SUFFIX WAS FOUND, CALL
;
;  IF LONGE FAILS OR THE ADDRESS RETURNED BY SUFFIX AND LONGE =
;  ADDRESS OF THE END OF THE WORD, OR IF THIS ADDRESS POINTS
;  TO THE LETTER H, OR IF THERE IS A VOWEL IN THE REMAINING
;  WORD STEM, THEN CONTINUE LOOKING FOR SUFFIXES,  ELSE
;  SKIP THAT PART OF THE ROUTINE.
;
        JM      FESUF           ;LONGE FAILED CHECK MORE SUFFIXES
        CALL    VOWEL           ;ANY VOWELS IN STEM?
        JP      FESUF           ;YES, CHECK MORE SUFFIXES
        POP     H               ;GET END ADDR FROM STACK
        PUSH    H               ;BUT LEAVE A COPY OF IT
        MOV     A,L             ;GET RIGHT PART OF END
        SUB     E               ;CMP WITH RIGHT OF RETURNED ADDR
        MOV     L,A             ;SAVE INTERMEDIATE RESULT
        MOV     A,H             ;GET LEFT OF END
        SBB     D               ;DOES RETURNED ADDR=END?
        ORA     L               ;COMBINE REST
        POP     H               ;RESTORE HL
        PUSH    H               ;BUT KEEP A COPY
        JZ      FESUF           ;YES, CHECK MORE SUFFIXES
        LDAX    D               ;GET LETTER POINTED TO
        CMI     150             ;IS IT AN H?
        JNZ     FEND2           ;NO, SKIP CHECK FOR SUFFIXES
;
FESUF:  POP     D               ;RESTORE OLD END VALUE
        PUSH    D               ;BUT KEEP A COPY ON STACK
;
;  CHECK REPETITIVELY FOR SUFFIXES IN SUFF1 STRING.  IF FOUND,
;  CALL INSERT TO INSERT THE BREAK MARK.
;
FELP:   LXI     H,SUFF1         ;GET ADDR OF SUFF1 STRING
        CALL    SUFFIX          ;ANY MATCHES?
        JM      FELPEX          ;NO, EXIT FROM LOOP
        POP     D               ;GET OLD END ADDR BACK
        PUSH    E               ;SAVE START ADDR
        PUSH    H               ;SAVE SUFFIX ADDR
        MOV     E,H             ;MOVE SUFFIX ADDR TO BC
        MOV     C,L             ;MOVE REST
        INX     E               ;ADD 1 TO SUFFIX ADDR
        CALL    INSERT          ;INSERT THE BREAK MARK
        POP     H               ;RESTORE SUFFIX ADDR
        POP     E               ;RESTORE START ADDR
        PUSH    D               ;RESAVE NEW END ADDR
```

```
        XCHG                    ;PUT SUFFIX ADDR IN DE
        JMP     FELP            ;LOOK FOR MORE SUFFIXES
;
;  NOW LOOK FOR A SUFFIX IN SUFF2.
;
FELPEX: LXI     H,SUFF2         ;GET ADDR OF SUFF2 STRING
        CALL    SUFFIX          ;ANY MATCHES?
        JM      FENO3           ;NO, CONTINUE
        POP     D               ;GET END ADDR BACK
        PUSH    B               ;SAVE START ADDR
        MOV     B,H             ;MOVE SUFFIX ADDR TO BC
        MOV     C,L             ;MOVE REST
        INX     B               ;ADD 1 TO SUFFIX ADDR
        CALL    INSERT          ;INSERT THE BREAK MARK
        POP     B               ;RESTORE BC
        RET                     ;RETURN FROM FINAL E
;
;  NOW CHECK FOR SUFFIXES IN SUFF3.    IF A SUFFIX IS FOUND,
;  AND IF THE SECOND LETTER IN THE SUFFIX IS AN E, RETURN, ELSE,
;  INSERT THE BREAK MARK AND CONTINUE.
;
FENO3:  LXI     H,SUFF3         ;GET ADDR OF SUFF3 STRING
        CALL    SUFFIX          ;ANY MATCHES?
        JM      FENO4           ;NO, SKIP TO NEXT TEST
        POP     D               ;GET OLD END ADDR BACK
        INX     H               ;GET ADDR OF 1ST CHAR IN SUFF
        INX     H               ;GET ADDR 2ND CHAR IN SUFF
        MOV     A,M             ;GET THE CHAR
        CMI     145             ;IS IT AN E?
        RZ                      ;IF YES, RETURN
        DCX     H               ;GET INSERT ADDR
        PUSH    B               ;SAVE START ADDR
        MOV     B,H             ;MOVE INSERT ADDR TO BC
        MOV     C,L             ;MOVE REST
        CALL    INSERT          ;INSERT THE BREAK MARK
        MOV     H,B             ;MOVE INSERT ADDR TO HL
        MOV     L,C             ;MOVE REST
        DCX     H               ;GET SUFFIX ADDR BACK
        POP     B               ;RESTORE BC
        PUSH    D               ;SAVE NEW END ADDR
        XCHG                    ;PUT SUFFIX ADDR IN DE
;
;   IF LAST LETTER IN WORD STEM IS AN I U OR Y, AND IT IS
;  THE ONLY VOWEL IN THE STEM, MARK IT LONG AND RETURN.
;
FENO4:  LDAX    D               ;GET LAST LETTER
        LXI     H,IUY           ;GET ADDR OF IUY
        CALL    CNEOF           ;IS IT ONE OF THEM?
        JM      FENO5           ;NO, CONTINUE
        CALL    VOWEL           ;ANY VOWELS?
        JP      FENO5           ;YES, SKIP TO NEXT TEST
```

```
        LDAX    D               ;GET THE CHAR
        ANI     337             ;MAKE UPPER CASE
        STAX    D               ;PUT IT BACK
        POP     D               ;GET END ADDR
        RET                     ;RETURN
;
;  IF FIRST LETTER IN SUFFIX IS AN A E OR O, RETURN.
;
FENO5:  MOV     H,D             ;MOVE SUFFIX ADDR TO HL
        MOV     L,E             ;MOVE REST
        INX     H               ;GET ADDR 1ST CHAR IN SUFF
        MOV     A,M             ;GET THE CHAR
        CMI     '?              ;IF THIS IS THE BREAK MARK, SKIP
        JNZ     FENO6           ;NO, USE IT
        INX     H               ;GET ADDR NEXT CHAR PAST BREAK
        MOV     A,M             ;GET THIS CHAR
FENO6:  LXI     H,AEIC          ;GET ADDR OF AEIO
        CALL    ONEOF           ;IS IT ONE OF THEM?
        JM      FEEXIT          ;IF NOT, RETURN
;
;  TRY LONG E AGAIN, IF FAILURE, RETURN
;
        CALL    LONGE           ;CALL LONG E PROCESSING
        JM      FEEXIT          ;IF FAILURE, EXIT
;
;  IF FIRST CHAR PAST INDEX ADDR IS A C OR G, AND THERE IS
;  A VOWEL IN THE WORD STEM, RETURN.
;
        LXI     H,CG            ;GET ADDR OF CG
        INX     D               ;GET ADDR 1 CHAR PAST INDEX
        LDAX    D               ;GET THE CHAR
        DCX     D               ;PUT INDEX BACK
        CALL    ONEOF           ;IS IT ONE OF THEM?
        JM      FENO7           ;NO, SKIP TO NEXT TEST
        CALL    VOWEL           ;ANY VOWELS?
        JP      FEEXIT          ;IF YES, RETURN
;
;  IF NEXT TWO CHARACTERS ARE TH, MAKE THEM VOICED TH.
;
FENO7:  INX     D               ;GET ADDR NEXT CHAR
        CALL    TH              ;ARE THEY TH?
        JM      FENO8           ;NO, SKIP TO NEXT TEST
        MVI     A,'T            ;GET A CAP T
        STAX    D               ;INSERT IN STRING
        INX     D               ;GET ADDR NEXT CHAR
        MVI     A,'H            ;GET A CAP H
        STAX    D               ;INSERT IN STRING
        DCX     D               ;PUT D BACK
FENO8:  DCX     D               ;PUT DE BACK TO INDEX VALUE
;
;  IF NEXT TWO CHARS ARE EI, RETURN
```

```
;
FENO2:   LDAX    D                  ;GET A CHAR
         CMI     145                ;IS IT AN E?
         JNZ     FENO9              ;NO, TRY NEXT TEST
         INX     D                  ;GET ADDR OF NEXT CHAR
         LDAX    D                  ;GET THE CHAR
         CMI     154                ;IS IT AN L?
         JZ      FEEXIT             ;IF YES, RETURN
         DCX     D                  ;PUT D BACK
;
;  IF INDEX=ADDR OF START OF WORD, OR FIRST CHAR BEFORE
;  INDEX IS NOT AN A, E OR O, MARK THE CHAR AT INDEX A LONG
;  VOWEL.
;
FENO9:   MOV     A,E                ;GET RIGHT OF INDEX
         SUB     C                  ;CMP WITH RIGHT OF START
         PUSH    H                  ;SAVE HL
         MOV     H,A                ;SAVE INTERMEDIATE RESULT
         MOV     A,D                ;GET LEFT OF INDEX
         SBB     B                  ;IS INDEX=START?
         ORA     H                  ;COMBINE WITH REST
         POP     H                  ;RESTORE HL
         JZ      FEUP               ;MARK AS LONG
         DCX     D                  ;GET ADDR PRECEEDING CHAR
         LDAX    D                  ;GET THE CHAR
         INX     D                  ;PUT D BACK
         LXI     H,AEO              ;GET ADDR OF AEO
         CALL    ONEOF              ;IS IT ONE OF THEM?
         JP      FEEXIT             ;IF YES, RETURN
FEUP:    LDAX    D                  ;GET THE CHAR
         ANI     337                ;MAKE UPPER CASE
         STAX    D                  ;PUT IT BACK
;
FEEXIT:  POP     D                  ;RESTORE DE
         RET                        ;RETURN
    ;
         .SBTTL  MIDDLE U PREPROCESSOR
;
;  THIS ROUTINE TESTS FOR A LONG MIDDLE U AND OTHER LONG
;  VOWELS APPEARING IN THE MIDDLE OF THE WORD.  ON ENTRY,
;   BC CONTAINS THE ADDRESS OF THE START OF THE WORD AND DE
;  CONTAINS THE ADDRESS OF THE END OF THE WORD.
;
MUADDR:  .BLKB   2                  ;INDEX SAVE AREA
MUPTR:   .BLKB   2                  ;POINTER TO THE TESTED VOWEL
;
MIDU:    PUSH    B                  ;SAVE BC
         PUSH    D                  ;SAVE DE
;  DO FOR I=START TO END-2.
;
         DCX     D                  ;GET END ADDR
```

```
        MOV     H,B             ;MOVE START TO HL
        MOV     L,C             ;MOVE REST
          DCX   H               ;GET STARTING INDEX VALUE
        SHLD    MUADDR          ;SAVE INDEX
;
MULP1:  LHLD    MUADDR          ;GET INDEX VALUE
        INX     H               ;ADD ONE
        MOV     A,L             ;GET RIGHT OF INDEX
        SUB     E               ;CMP WITH RIGHT OF END
        MOV     A,H             ;GET LEFT OF INDEX
        SBB     D               ;IS INDEX>=END?
        JNC     MUL1EX          ;YES, EXIT FROM LOOP
;
        SHLD    MUADDR          ;SAVE NEW INDEX
;
;  IF THE CHARACTER AT THE INDEX IS NOT A U, SKIP TO NEXT LETTER.
;
        MOV     A,M             ;GET THE LETTER
        CMI     165             ;IS IT A U?
        JNZ     MULP1           ;NO, SKIP
;
        MOV     B,H             ;MOVE INDEX TO BC
        MOV     C,L             ;MOVE REST
;
;  IF THE U IS NOT PRECEEDED BY AN AEIOU AND FOLLOWED BY
;  NOT AN AEIOUWXY OR BCDFGKPT THEN R, WHICH IS FOLLOWED BY AN
;  AEIOUY, THEN THE U IS LONG.
;
        DCX     B               ;GET ADDR PRECEEDING CHAR
        LDAX    B               ;GET THE CHAR
        LXI     H,AEIOU         ;GET ADDR AEIOU
        CALL    ONEOF           ;IS IT ONE OF THEM?
        JZ      MULP1           ;IF YES, FAIL
;
        INX     B               ;PUT INDEX BACK
        INX     B               ;GET ADDR FOLLOWING CHAR
        LDAX    B               ;GET THE CHAR
        LXI     H,AEIWXY        ;GET ADDR AEIOUWXY
        CALL    ONEOF           ;IS IT ONEOF THEM?
        JZ      MULP1           ;IF YES, FAIL
;
        INX     B               ;GET ADDR NEXT CHAR
        LDAX    B               ;GET THE CHAR
        CMI     162             ;IS IT AN R?
        JNZ     MUSKIP          ;IF NOT, SKIP TO NEXT TEST
;
        DCX     B               ;GET ADDR PREVIOUS CHAR
        LDAX    B               ;GET THE CHAR
        INX     B               ;PUT INDEX BACK
        LXI     H,BCDFGK        ;GET ADDR BCDFGKPT
        CALL    ONEOF           ;IS IT ONE OF THEM?
```

```
        JM      MUSKIP          ;IF NOT, SKIP ADJUSTMENT
        INX     B               ;TREAT COMBINATION AS ONE LETTER.
MUSKIP: LDAX    B               ;GET NEXT CHAR
        ORI     40              ;MAKE LOWER CASE
        LXI     H,AEICUY        ;GET ADDR OF AEIOUY
        CALL    CNEOF           ;IS IT ONE OF THEM?
        JM      MULP1           ;IF NOT, FAIL
;
;  MARK THE U AS LONG BY UPPER CASE.
;
        LHLD    MUADDR          ;GET ADDR OF U
        MVI     M,'U            ;INSERT A CAP U
        JMP     MULP1           ;GO TRY FOR MORE
;
;  NOW DO FOR I=START TO END-3.
;
MUL1EX: POP     D               ;GET END VALUE BACK
        POP     H               ;GET START ADDR BACK
        PUSH    H               ;KEEP COPY ON STACK
        PUSH    D               ;KEEP COPY ON STACK
        DCX     D               ;SET END TO END-2
        DCX     D               ;FINISH END=END-2
        DCX     H               ;SET STARTING INDEX VALUE
        SHLD    MUADDR          ;SAVE INDEX
;
MULP2:  LHLD    MUADDR          ;GET INDEX VALUE
        INX     H               ;ADD ONE
        MOV     A,L             ;GET RIGHT OF INDEX
        SUB     E               ;CMP WITH RIGHT OF END
        MOV     A,H             ;GET LEFT OF INDEX
        SBB     D               ;IS INDEX>=END?
        JNC     MUL2EX          ;IF YES, EXIT FROM LOOP
;
        SHLD    MUADDR          ;SAVE INDEX VALUE
        SHLD    MUPTR           ;SAVE AGAIN
        MOV     E,H             ;MOVE INDEX TO BC
        MOV     C,L             ;MOVE REST
;
;  IF THE CHAR AT THE INDEX IS NOT AN A E OR C, SKIP IT
        LDAX    B               ;GET THE CHAR
        LXI     H,AEC           ;GET ADDR AEO
        CALL    CNEOF           ;IS IT ONE OF THEM?
        JM      MULP2           ;NO, SKIP
;
;  IF THE AEC IS FOLLOWED BY AEIOUWXY? THEN FAIL
;
        INX     B               ;GET ADDR NEXT CHAR
        LDAX    B               ;GET THE CHAR
        LXI     H,AEIXYQ        ;ADDR OF AEIOUWXY?
        CALL    CNEOF           ;IS IT ONE OF THEM?
        JZ      MULP2           ;IF SO, FAIL
```

```
;
;;  IF THE NEXT TWO LETTERS ARE TH, TREAT THEM AS ONE LETTER.
;
        MOV     H,B             ;MOVE INDEX ADDR TO HL
        MOV     L,C             ;MOVE REST
        CALL    TH              ;IS IT TH?
        JM      MUSKP2          ;IF NOT, SKIP ADJUSTMENT
        LHLD    MUADDR          ;GET INDEX
        INX     H               ;ADD 1
        SHLD    MUADDR          ;PUT IT BACK
        INX     B               ;GET INDEX PAST TH
;
;  IF BCDFGKPT FOLLOWED BYR THEN I  IS NEXT, ADD 1  TO INDEX
;
MUSKP2: INX     B               ;GET ADDR NEXT CHAR
        LDAX    B               ;GET THE CHAR
        CMI     162             ;IS IT AN R?
        JNZ     MUSKP3          ;IF NOT, SKIP TO NEXT TEST
        INX     B               ;GET ADDR NEXT CHAR
        LDAX    B               ;GET THE  CHAR
        CMI     151             ;IS IT AN I?
        JNZ     MUSKP3          ;IF NOT, SKIP TO NEXT TEST
        DCX     B               ;GET ADDR OF THE R
        DCX     B               ;GET ADDR PREVIOUS CHAR
        LDAX    B               ;GET THE CHAR
        LXI     H,BCDFGK        ;GET ADDR OF BCDFGKPT
        CALL    CNECF           ;IS IT ONE OF THEM?
        JM      MUSKP3          ;IF NOT, SKIP ADJUSTMENT
;
        LHLD    MUADDR          ;GET INDEX VALUE
        INX     H               ;ADD ONE
        SHLD    MUADDR          ;SAVE INDEX
;
;  IF AN I OR E IS NEXT FOLLOWED BY AN A O OR U, THE ORIGINAL
;  A E OR O IS LONG
;
MUSKP3: LHLD    MUADDR          ;GET INDEX
        INX     H               ;ADD ONE
        INX     H               ;ADD ONE MORE
        MOV     B,H             ;MOVE TO BC
        MOV     C,L             ;MOVE REST
        LDAX    B               ;GET THE CHAR
        LXI     H,IE            ;GET ADDR IE
        CALL    CNEOF           ;IS IT ONE OF THEM?
        JM      MUSKP4          ;IF NOT, SKIP TO NEXT TEST
;
        INX     B               ;GET ADDR NEXT CHAR
        LDAX    B               ;GET THE CHAR
        LXI     H,AOU           ;GET ADDR AOU
        DCX     B               ;PUT INDEX BACK
        ORI     40              ;MAKE LOWER CASE
```

```
        CALL    CNECF           ;IS IT ONE OF THEM?
        JZ      MUMARK          ;YES, MARK IT
;
;  IF NEXT IS AN E THEN I THEN N, MARK ORIGINAL LETTER LONG.
;
MUSKP4: LDAX    E               ;GET THE CHAR
        CMI     145             ;IS IT AN E?
        JNZ     MULP2           ;IF NOT, FAIL
;
        INX     E               ;GET ADDR NEXT CHAR
        LDAX    B               ;GET THE CHAR
        CMI     151             ;IS IT AN I?
        JNZ     MULP2           ;IF NOT, FAIL
;
        INX     E               ;GET ADDR NEXT CHAR
        LDAX    B               ;GET THE CHAR
        CMI     156             ;IS IT AN N?
        JNZ     MULP2           ;IF NOT, FAIL
;
;  MARK THE LETTER AS LONG.
;
MUMARK: LHLD    MUPTR           ;GET POINTER TO LETTER
        MOV     A,M             ;GET THE LETTER
        ANI     337             ;MARK UPCASE
        MOV     M,A             ;PUT IT BACK
        JMP     MULP2           ;GO TRY FOR MORE
;
;  GET START AND END OF WORD ADDRESSES BACK.  THEN CHECK FOR
;  LONG I AND Y.
;
MUL2EX: POP     D               ;GET END ADDR BACK
        POP     E               ;GET START ADDR BACK
        PUSH    E               ;KEEP START ADDR COPIED ON STACK
;
;  IF FIRST LETTER IS Y, IGNORE IT.
;
        LDAX    B               ;GET FIRST CHAR
        CMI     171             ;IS IT A Y?
        JNZ     MUSKP5          ;YES, SKIP ADJUSTMENT
MULP3:  INX     B               ;SKIP PAST THE CHAR TO NEXT
;
;  DO FOR I=START TO END WHILE CHAR AT I IS NOT ONE OF AEIOUY.
;
MUSKP5: MOV     A,C             ;GET RIGHT OF INDEX
        SUB     E               ;CMP WITH RIGHT OFF END
        MOV     A,B             ;GET LEFT OF INDEX
        SBB     D               ;IS INDEX>=END?
        JNC     MUL3EX          ;IF SO, EXIT
;
        LDAX    B               ;GET THE CHAR
        ORI     40              ;MAKE LOWER CASE
```

```
        LXI     H,AEICUY            ;GET ADDR AEIOUY
        CALL    CNECF               ;IS IT ONE OF THEM?
        JM      MULP3               ;KEEP LOOKING
;
;  IF THE VOWEL IS NOT I CR Y, RETURN.
;
MUL3EX: LDAX    B                   ;GET THE CHAR
        LXI     H,IY                ;GET ADDR OF IY
        CALL    CNEOF               ;IS IT ONE OF THEM?
        JM      MUEXIT              ;IF NCT, RETURN
;
;   IF THE NEXT LETTER IS AN A O OR U, THE I OR Y IS LONG.
;
        INX     B                   ;GET ADDR NEXT CHAR
        LDAX    B                   ;GET THE CHAR
        LXI     H,AOU               ;GET ADDR AOU
        CALL    CNEOF               ;IS IT ONE OF THEM?
        JM      MUEXIT              ;NO, RETURN
;
;  MARK IT LONG BY UPCASE.
;
        DCX     B                   ;GET ADDR OF I OR Y BACK
        LDAX    B                   ;GET THE CHAR
        ANI     337                 ;MAKE UPCASE
        STAX    B                   ;PUT IT BACK
;
MUEXIT: POP     B                   ;RESTORE START ADDR
        RET                         ;RETURN
;
        .SBTTL  MIDDLE E PREPROCESSOR
;
;  THIS ROUTINE PREPROCESSES MIDDLE E'S IN AN ATTEMPT TO
;  DETERMINE IF THEY ARE LONG OR SILENT E'S.  ON ENTRY, BC
;  CONTAINS THE ADDRESS OF THE BEGINNING OF THE WORD TO BE
;  PROCESSED AND DE CONTAINS THE ADDRESS OF THE END OF THE WORD.
;
MEEND:  .BLKB   2                   ;STOPPING VALUE FOR LLOOP
MESW:   .BLKB   1                   ;SWITCH TO COUNT CHARACTERS
;
MIDE:   PUSH    B                   ;SAVE START ADDR
        XRA     A                   ;GET A ZERO
        STA     MESW                ;SET SWITCH TO ZERO
;  CALCULATE END VALUE FOR LOOP AND STORE IN MEEND.
;
        MOV     H,D                 ;PUT END OF WORD ADDR IN HL
        MOV     L,E                 ;MOVE REST
        DCX     H                   ;SET END TO END-2
        DCX     H                   ;FINISH END=END-2
        SHLD    MEEND               ;STORE IN MEEND
        INX     B                   ;SET START ADDR TO SSTART+2
        INX     B                   ;FINISH START=START+2
```

```
;
;   DO THE FOLLOWING LOOP AS I GOES FROM START+3 TO END-3.
;
MELP:   INX     B               ;ADD 1 TO INDEX
        MOV     A,C             ;GET RIGHT OF INDEX
        SUB     L               ;CMP WITH RIGHT OF END
        MOV     A,B             ;GET LEFT OF INDEX
        SBB     H               ;IS INDEX >= END?
        JNC     MEEXIT          ;EXIT FROM ROUTINE
;
;   INCREMENT CHARACTER COUNT.
;
        LXI     H,MESW          ;GET ADDR OF COUNT
        INR     M               ;ADD ONE
;
;   IF THE CHARACTER AT THE INDEX IS NOT AN E, TRY NEXT CHARACTER.
;
        LDAX    B               ;GET THE CHARACTER
        CMI     145             ;IS IT AN E?
        JNZ     MELP            ;NO, SKIP IT
;
;
        PUSH    B               ;SAVE INDEX VALUE
;
;   IF COUNT IS LESS THAN 3, SKIP THIS TEST.
;
        LDA     MESW            ;GET THE COUNT
        CMI     3               ;IS IT < 3?
        JC      MENO            ;YES, SKIP THIS TEST
;
;IF A B C D F G M N P R S OR T IS FOLLOWED BY B D F G K P OR T
;   WHICH IS FOLLOWED BY AN L WHICH IS FOLLOWED BY AN E
;   AND THERE IS AT LEAST ONE SYLLABLE AFTER THE E, IT IS A
;   A SILENT E AND SHOULD BE SO MARKED.
;
        INX     B               ;GET ADDR OF CHAR AFTER E
        LXI     H,AEICUQ        ;ADDR OF AEIOUY?
        CALL    SYLTEST         ;IS IT A SYLLABLE?
        JM      MENO            ;NO, SKIP TO NEXT TEST
;
        DCX     B               ;PUT INDEX BACK
        DCX     B               ;GET ADDR OF CHAR LEFT OF E
        LDAX    B               ;GET THE CHAR
        CMI     154             ;IS IT AN L?
        JNZ     MENO            ;NO, SKIP TO NEXT TEST
;
        DCX     B               ;GET ADDR NEXT CHAR LEFT
        LDAX    B               ;GET THE CHAR
        LXI     H,BDFGKP        ;ADDR OF BDFGKPT
        CALL    ONEOF           ;IS IT ONEOF THEM?
        JM      MENO            ;NO, SKIP TO NEXT TEST
```

```
;
        DCX     B               ;GET ADDR NEXT CHAR LEFT
        LDAX    B               ;GET THE CHAR
        LXI     H,BCDFGM        ;ADDR OF BCDFGMMPRST
        CALL    CNEOF           ;IS IT ONEOF THEM?
        JP      MEMARK          ;YES, MARK THE SILENT E
;
;   IF NOT AN AEIU IS FOLLOWED BY AN AIOUY WHICH IS FOLLOWED BY NOT
;  AN AEHIOUWXY WHICH IS FOLLOWED BY AN E AND THERE IS AT
;  LEAST ONE SYLLABLE TO THE RIGHT OF THE E, THE E IS SILENT
;  AND SHOULD BE SO MARKED.
;
;  IF AN E IS FOUND WHICH FAILS BOTH OF THESE TESTS, MIDDLE
;  E PROCESSING IS TERMINATED.
;
MENO:   POP     B               ;GET THE INDEX BACK
        PUSH    B               ;BUT SAVE A COPY
        LXI     H,AEINOR        ;ADDR OF AEINORUY?
        INX     B               ;ADD ONE
        CALL    SYLTEST         ;IS THERE A SYLLABLE?
        JM      MENO2           ;NO, RETURN
;
        DCX     B               ;GET ADDR OF E BACK
        DCX     B               ;GET ADDR NEXT CHAR LEFT
        LDAX    B               ;GET THE CHAR
        LXI     H,AEHIOU        ;ADDR OF AEHIOUWXY
        CALL    CNEOF           ;IS IT ONEOF THEM?
        JP      MENO2           ;YES, RETURN
;
        DCX     B               ;GET ADDR NEXT CHAR LEFT
        LDAX    B               ;GET THE CHAR
        LXI     H,AIOUYU        ;GET ADDR OF AIOUYU
        CALL    CNEOF           ;IS IT ONEOF THEM?
        JM      MENO2           ;NO, RETURN
;
        DCX     B               ;GET ADDR OF NEXT CHAR LEFT
        LDAX    B               ;GET THE CHAR
        LXI     H,AEIU          ;GET ADDR OF AEIU
        CALL    CNEOF           ;IS IT ONE OF THEM?
        JP      MENO2           ;YES, RETURN
;
;  IF THIS LETTER (POINTED TO BY INDEX) IS NOT AN O, MARK
;  THE NEXT LETTER AS A LONG VOWEL.
;
        LDAX    B               ;GET THE CHAR
        CMI     156             ;IS IT AN O?
        JZ      MEMARK          ;NO, SKIP THE MARKING
        INX     B               ;GET ADDR OF NEXT CHAR
        LDAX    B               ;GET THE CHAR
        ANI     337             ;MAKE UPCASE
        STAX    B               ;PUT IT BACK
```

```
;
;  MARK SILENT E'S BY CALLING INSERT ROUTINE.
;
MEMARK:  POP     B              ;GET INDEX VALUE BACK
         INX     B              ;GET ADDR FOR MARK
         CALL    INSERT         ;INSERT THE MARK
         LHLD    MEEND          ;GET AEND VALUE FOR LOOP
         INX     H              ;EXTENT BY ONE
         SHLD    MEEND          ;PUT IT BACK
         JMP     MELP           ;GO LOOK FOR MORE E'S
;
MENO2:   POP     B              ;REMOVE OLD INDEX VALUE
MEEXIT:  POP     B              ;RESTORE START ADDR
         RET                    ;RETURN
;
         .SBTTL  MIDDLE S PREPROCESSOR
;
;   THIS ROUTINE DETERMINES IF A MIDDLE S SHOULD BE VOICED OR
;  UNVOICED.  ON ENTRY, BC CONTAINS THE ADDR OF THE BEGINNING
;  OF THE WORD AND DE CONTAINS THE ADDRESS OF THE END.
;
MIDS:    PUSH    B              ;SAVE STARTING ADDR
;
;  LOOP THROUGH THE CHARACTERS OF THE WORD.
;
MSLP:    INX     B              ;GET ADDR NEXT CHAR
         MOV     A,C            ;GET RIGHT OF INDEX
         SUB     E              ;CMP WITH RIGHT OF END
         MOV     A,B            ;GET LEFT OF INDEX
         SBB     D              ;IS INDEX>=END?
         JNC     MSEXIT         ;YES, DONE
;
;  IF THE CHAR IS NOT AN S, SKIP TO NEXT CHAR.
;
         LDAX    B              ;GET THE CHAR
         CMI     163            ;IS IT AN S?
         JNZ     MSLP           ;IF NOT, SKIP
;
;  IF AN S IS PRECEEDED BY AN AEIOUY AND FOLLOWED BY AN
;  AEIMOUY IT IS A VOICED S AND SHOULD BE SO MARKED.
;
         DCX     B              ;GET ADDR CHAR BEFORE S
         LDAX    B              ;GET THE CHAR
         INX     B              ;PUT INDEX BACK
         ORI     40             ;MAKE LOWER CASE
         LXI     H,AEIOUY       ;GET ADDR AEIOUY
         CALL    ONEOF          ;IS IT ONE OF THEM?
         JM      MSLP           ;NO, SKIP IT
;
         INX     B              ;GET ADDR CHAR AFTER S
         LDAX    B              ;GET THE CHAR
```

```
        ORI     40                  ;MAKE LOWER CASE
        DCX     B                   ;PUT INDEX BACK
        LXI     H,AEIMOU            ;GET ADDR OF AEIMOUY
        CALL    CMEOF               ;IS IT ONE OF THEM?
        JM      MSLP                ;NO, SKIP IT
;
        MVI     A,'S                ;GET A CAP S
        STAX    B                   ;INSERT IN WORD
        JMP     MSLP                ;LOOK FOR MORE
;
MSEXIT: POP     B                   ;RESTORE BC
        RET                         ;RETURN
;
        .SBTTL  SEARCH FOR SUFFIX
;
;  THIS ROUTINE CHECKS THE END OF THE INPUT STRING FOR ANY
;  OF THE SUFFIXES IN THE SUFFIX STRING.  ON ENTRY, BC
;  CONTAINS THE ADDRESS OF THE START OF THE INPUT STRING,
;  DE CONTAINS THE ADDRESS OF THE END OF THE INPUT STRING,
;  AND HL CONTAINS THE STRING OF SUFFIXES TO BE CHECKED.
;  IF A SUFFIX IS FOUND, HL RETURNS WITH THE ADDRESS WITHIN
;  THE INPUT WORD WHERE THE SUFFIX STARTS AND THE ACC RETURNS
;  WITH 0.  IF NO SUFFIX IS FOUND, THE ACC RETURNS WITH -1.
;
SUFFIX: PUSH    D                   ;SAVE END OF STRING ADDR
;
;  FIRST, SEE IF SUFFIX STRING IS EXHAUSTED.  IF SO, RETURN.
;
        MOV     A,M                 ;GET NEXT SUFFIX CHAR
        ORA     A                   ;IS IT THE MARK?
        JZ      SUNO                ;YES, RETURN NO SUFFIX
;
;  SEE IF SUFFIX CHARACTERS MATCH THOSE OF END OF WORD.
;
SULP1:  LDAX    D                   ;GET A CHAR FROM WORD
        CMP     M                   ;DOES IT MATCH SUFFIX?
        JNZ     SUEND               ;NO, END OF MATCHES
        INX     H                   ;GET ADDR NEXT SUFFIX CHAR
        DCX     D                   ;GET ADDR NEXT WORD CHAR
        JMP     SULP1               ;KEEP CHECKING
;
;  SEE IF SUFFIX IS A COMPLETE ONE.
;
SUEND:  MOV     A,M                 ;GET THE SUFFIX CHAR
        ORA     A                   ;IS IT THE MARK?
        JNZ     SUSKIP              ;NO, NOT A MATCH, SKIP IT
;
;  COMPLETE MATCH.  IF REMAINING WORD STEM CONTAINS NO VOWELS
;  SUFFIX IS NOT A VALID ONE.  ABORT SEARCH AND RETURN WITH
;  NO SUFFIXES.
;
```

```
        INX     D                       ;INCREASE LENGTH OF WORD
        CALL    VOWEL                   ;ANY VOWELS?
        JM      SUNO                    ;NO, RETURN NO
;
;  RETURN YES AND ADDR OF SUFFIX.
;
        DCX     D                       ;PUT END ADDR BACK
        XCHG                            ;PUT SUFFIX ADDR IN HL
        POP     D                       ;RESTORE DE
        XRA     A                       ;SET SUCCESS FLAG
        RET                             ;RETURN
;
;  IF SUFFIX DID NOT MATCH, RESTORE END OF WORD ADDR, SKIP
;  REST OF CURRENT SUFFIX, AND TRY SEARCH AGAIN.
;
SUSKIP: POP     D                       ;RESTORE END ADDR
        XRA     A                       ;GET A MARK CHAR
SULP2:  CMP     M                       ;IS THIS CHAR A MARK?
        JZ      SUNEXT                  ;YES, CHECK NEXT SUFFIX
        INX     H                       ;GET ADDR NEXT CHAR
        JMP     SULP2                   ;CHECK NEXT  CHAR
;
;  SKIP PAST THE MARK AND RETRY SEARCH.
;
SUNEXT: INX     H                       ;SKIP MARK
        JMP     SUFFIX                  ;TRY AGAIN
;
;  IF NO SUFFIX FOUND, RETURN FLAG INDICATING SO.
;
SUNO:   ORI     -1                      ;SET NO FLAG
        POP     D                       ;RESTORE DE
        RET                             ;RETURN
SUFF0:  .ASCIZ  /la/
        .ASCIZ  /el/
        .ASCIZ  /o/
        .ASCIZ  /er/
        .ASCIZ  /su/
        .ASCIZ  /y/
        .BYTE   0
SUFF1:  .ASCIZ  /elba/
        .ASCIZ  /de/
        .ASCIZ  /re/
        .ASCIZ  /gni/
        .ASCIZ  /tse/
        .ASCIZ  /ne/
        .ASCIZ  /ylba/
        .ASCIZ  /yl/
        .ASCIZ  /ro/
        .ASCIZ  /yre/
        .ASCIZ  /ye/
        .ASCIZ  /ssel/
```

```
        .ASCIZ  /ssen/
        .ASCIZ  /luf/
        .ASCIZ  /tnem/
        .BYTE   0
SUFF2:  .ASCIZ  /ci/
        .ASCIZ  /laci/
        .BYTE   0
SUFF3:  .ASCIZ  /e/
        .BYTE   0
;
        .SBTTL  INSERT THE BREAK MARK
;
;  THIS ROUTINE INSERTS THE BREAK MARK (?) IN A STRING OF
;  CHARACTERS.  ON ENTRY, BC CONTAINS THE ADDRESS WHERE THE
;  MARK SHOULD BE PLACED AND DE CONTAINS THE ADDRESS OF THE
;  END OF THE STRING.  THE ADDRESS OF THE NEW END OF THE
;  STRING IS RETURNED IN DE.
;
INSERT: PUSH    B               ;SAVE STARTING ADDR
;
;  IF THE BREAK POINT CONTAINS AN E, PUT BREAK MARK AFTER IT.
;
        LDAX    B               ;GET FIRST CHAR
        CMI     145             ;IS IT AN E?
        JNZ     INNOTE          ;NO, NOT AN E
;
;  SEE IF A BREAK MARK ALREADY EXISTS AFTER THE E.  IF SO, RETURN.
;
        INX     B               ;GET ADDR OF NEXT CHAR
        LDAX    B               ;GET THE CHAR
        CMI     '?              ;IS IT THE BREAK MARK?
        JZ      INEXIT          ;YES, RETURN
;
;  STARTING FROM RIGHT TO LEFT, MOVE EACH CHARACTER ONE POSITION
;  TO THE RIGHT.
;
INNOTE: INX     D               ;INCREASE LENGTH
        MOV     H,D             ;COPY END ADDR TO WORK REG
        MOV     L,E             ;COPY REST
;
;  IF INDEX (HL) < START OF STRING (BC), TRANSFER DONE.
;
INLP:   MOV     A,L             ;GET RIGHT PART OF INDEX
        SUB     C               ;CMP WITH IN ADDR
        MOV     A,H             ;GET REST OF  INDEX
        SBB     B               ;IS INDEX<START?
        JC      INMARK          ;YES, PLACE THE BREAK
;
        MOV     A,M             ;GET A CHAR
        INX     H               ;GET DESTINATION ADDR
        MOV     M,A             ;MOVE THE CHAR
```

```
        DCX     H               ;PUT INDEX BACK
        DCX     H               ;GET ADDR NEXT LEFT CHAR
        JMP     INLP            ;GO MOVE IT
;
;  INSERT THE BREAK IN THE SPACE JUST MADE AVAILABLE.
;
INMARK: MVI     A,'?            ;LOAD THE BREAK MARK
        STAX    B               ;INSERT IN STRING
;
INEXIT: POP     B               ;RESTORE BC
        RET                     ;RETURN
;
        .SBTTL  CNEOF, CHECK FOR BEING ONE OF A STRING
;
;  THIS ROUTINE CHECKS TO SEE IF THE CHARACTER PASSED IN THE
;  ACC IS ONE OF A STRING OF CHARACTERS WHOSE ADDRESS IS PASSED
;  IN HL.  IF SO, 0 IS RETURNED IN THE ACC.  IF NOT, -1 IS
;  RETURNED.
;
CNEOF:  PUSH    B               ;SAVE BC
        MOV     B,A             ;SAVE THE MATCH CHAR
CCLP:   MOV     A,M             ;GET A CHAR FROM STRING
        CMA     A               ;IS THIS THE END OF THE STRING?
        JZ      CCNO            ;YES, RETURN NO
;
        INX     H               ;POINT TO NEXT CHAR
        CMP     B               ;DOES THE CHAR MATCH?
        JNZ     CCLP            ;NO, KEEP LOOKING
;
;  MATCH FOUND, RETURN YES.
;
        XRA     A               ;SET YES FLAG
        POP     B               ;RESTORE BC
        RET                     ;RETURN
;
;  MATCH NOT FOUND, RETURN NO.
;
CONC:   ORI     -1              ;SET NO FLAG
        POP     B               ;RESTORE BC
        RET                     ;RETURN
;
        .SBTTL  SEE IF A STRING CONTAINS A VOWEL
;
;   THIS ROUTINE CHECKS A CHARACTER STRING TO DETERMINE IF
;  IT CONTAINS A VOWEL.  ON ENTRY, THE ADDRESS OF THE CHARACTER
;  STRING IS PASSED IN BC, DE CONTAINS THE ADDRESS OF THE
;  END OF THE CHARACTER STRING.  IF A VOWEL IS FOUND, 0 IS
;  RETURNED IN THE ACC.  IF NOT, -1 IS RETURNED.
;
VOWEL:  PUSH    D               ;SAVE DE
VOLP:   DCX     D               ;DECREMENT INDEX
```

```
;
;  IF WORD END IS LESS THAN WORD START, SEARCH IS OVER,
;  RETURN NO.
;
        MOV     A,E             ;GET RIGHT PART OF END
        SUB     C               ;CMP WITH RIGHT OF START
        MOV     A,D             ;GET LEFT OF END
        SBB     E               ;IS END<START?
        JNC     VCMORE          ;NO, KEEP CHECKING
;
        ORI     -1              ;SET NC FLAG
        POP     D               ;RESTORE DE
        RET                     ;RETURN
;
;  SEE IF THE NEXT CHARACTER IS A VOWEL.
;
VCMORE: LDAX    D               ;GET A CHARACTER
        ORI     40              ;MAKE LOWER CASE
        LXI     H,AEICUY        ;GET ADDR OF VOWELS
        CALL    CNEOF           ;IS IT A VOWEL?
        JM      VCLP            ;NO, KEEP CHECKING
;
;  RETURN WITH YES FLAG IN ACC.
;
        POP     D               ;RESTORE DE
        RET                     ;RETURN
;
        .SBTTL  SYLLABLE TEST
;
;  THIS ROUTINE TESTS A STRING OF CHARACTERS TO SEE IF IT
;  IS A SYLLABLE.  ON ENTRY, BC CONTAINS THE ADDRESS OF THE
;  START OF THE STRING, DE CONTAINS THE ADDRESS OF THE END
;  OF THE STRING, AND HL CONTAINS THE ADDRESS OF A STRING
;  OF EXCEPTION CHARACTERS.IF THE INPUT STRING IS A SYLLABLE,
;  0 IS RETURNED IN THE ACC, IF NOT, -1 IS RETURNED.
;
SYLTEST: PUSH   B               ;SAVE BC
        PUSH    D               ;SAVE DE
;
;  IF THE FIRST CHARACTER OF THE INPUT STRING IS IN THE
;  EXCEPTION STRING, THE STRING IS NOT A SYLLABLE.  RETURN
;  WITH THE NO FLAG.
;
        LDAX    E               ;GET THE FIRST CHAR
        ORI     40              ;MAKE LOWER CASE
        CALL    CNEOF           ;IS IT IN EXCEPTION STRING?
        JP      STNO            ;YES, RETURN NO
;
;  CHECK CHARACTERS IN INPUT STRING FROM LEFT TO RIGHT.  IF A BREAK
;  MARK (?) OR AN E FOLLOWED BY A BREAK IS FOUND, RETURN NO.
;  IF A VOWEL IS FOUND, RETURN YES.
```

```
;
STLP:   INX     E               ;PCINT TO NEXT INPUT CHAR
        MOV     A,C             ;GET RIGHT OF INDEX ADDR
        SUB     E               ;CMP WITH RIGHT OF END ADDR
        MCV     A,B             ;GET REST OF INDEX
        SBB     D               ;IS INDEX>=END ADDB?
        JNC     STNO            ;IF SO, NO SYLLABLE
;
        LDAX    E               ;GET A CHARACTER
        CMI     '?              ;IS IT THE BREAK MARK?
        JZ      STNO            ;YES, NO SYILABLE
;
        CMI     145             ;IS IT AN E?
        JNZ     STNOTE          ;NC, NOT AN E
;
        INX     B               ;GET ADDR NEXT CHAR
        LDAX    B               ;LOAD NEXT CHAR
        CMI     '?              ;IS IT FOLLOWED BY A BREAK?
        JZ      STNO            ;YES, NOT A SYLLABLE
;
        DCX     B               ;PUT INDEX POINTER BACK
STNOTE: LXI     H,AEICUY        ;GET ADDB OF VOWELS
        CRI     40              ;MAKE CHAR LOWER CASE
        CALI    CNECF           ;IS IT A VCWEI?
        JM      STLP            ;NO, KEEP CHECKING
;
        JMP     STEXIT          ;YES, RETURN WITH YES FLAG
;
STNO:   ORI     -1              ;SET NO FLAG
STEXIT: POP     D               ;RESTORE DE
        PCP     B               ;RESTORE BC
        RET                     ;RETURN
;
        .SBTTL  TH,     SEE IF A STRING = 'TH'
;
;  THIS ROUTINE CHECKS THE CHARACTERS POINTED TC BY HL TC
;  DETERMINE IF THEY EQUAL 'TH'.  IF SC, 0 IS RETURNED IN THE
;  ACC.  IF NCT, -1 IS RETURNED,.
;
TH:     MOV     A,M             ;GET FIRST CHAR
        CMI     154             ;IS IT  A T?
        JNZ     THNO            ;NO, RETURN NO FLAG
        INX     H               ;GET ADDR SECCND CHAR
        MOV     A,M             ;GET SECOND CHAR
        DCX     H               ;PUT HL BACK
        SUB     150             ;IS IT H?
        RZ                      ;IF YES, RETURN WITH 0
;
THNO:   ORI     -1              ;SET NO FLAG
        RET                     ;RETURN
;
```

```
        .SBTTL  LONG E TEST
;
;  THIS ROUTINE PERFORMS A TEST TO SEE IS AN E IS LONG.
;  ON ENTRY DE CONTAINS THE ADDRESS OF THE LETTER TO BE CHECKED.
;  IF NO LONG E IS FOUND, -1 IS RETURNED IN THE ACC.  IF
;  A LONG E IS FOUND, A 0 IS RETURNED AND THE ADDRESS OF THE
;  E IS RETURNED IN DE.
;
LONGE:  PUSH    D               ;SAVE DE
;
;  IF LAST TWO LETTERS ARE TH, SKIP THEM.
;
        MOV     H,D             ;MOVE ADDR TO HL
        MOV     L,E             ;MOVE REST
        DCX     H               ;GET ADDR OF LAST TWO LRS
        CALL    TH              ;ARE THEY TH?
        JM      LENO            ;NO, LEAVE END ADDR ALONE
        DCX     D               ;SKIP TH
;
;  IF DE POINTS TO A  A E I O U W X Y THEN LONG E FAILS
;
LENO:   LDAX    D               ;GET THE CHAR
        LXI     H,AEIWXY        ;GET ADDR OF AEIOUWXY
        ORI     40              ;MAKE LOWER CASE
        CALL    CNEOF           ;IS IT ONEOF THEM?
        JP      LEFAIL          ;IF YES, FAIL
;
;  IF PRECEEDING CHAR IS AN A E I O U Y, THEN LONG E SUCCEEDS.
;
        DCX     D               ;GET ADDR NEXT CHAR LEFT
        LDAX    D               ;GET THE CHAR
        LXI     H,AEIOUY        ;GET ADDR OF AEIOUY
        CALL    CNEOF           ;IS IT ONE OF RHEM?
        JM      LEFAIL          ;NO, FAILURE
;
        POP     H               ;REMOVE OLD DE FROM STACK
        RET                     ;RETURN SUCCESSFULLY
;
LEFAIL: POP     D               ;RESTORE OLD DE VALUE
        ORI     -1              ;SET FAILURE CODE
        RET                     ;RETURN
;
        .SBTTL  MOVE A STRING IN MEMORY
;
;  THE FOLLOWING MOVES A STRING OF CHARACTERS FROM A LOCATION WHOSE
;  ADDRESS IS IN THE HL REGISTERS TO THE LOCATION
;  STARTING AT THE ADDRESS IN THE DE REGISTERS.  THE MOVE CONTINUES
;  UNTIL A BYTE OF ZERO HAS BEEN MOVED.
;  ON ENTRY, HL CONTAINS THE ADDRESS OF THE SOURCE STRING,
;  DE CONTAINS THE ADDRESS OF THE DESTINATION.
;  ON EXIT, HL POINTS TO THE FIRST CHARACTER PAST THE BYTE OF  ZERO,
```

```
;  DE POINTS TO THE BYTE OF ZERO IN THE DESTINATION STRING.
;
MOVSAV:  .BYTE   C                  ;SAVE SPACE FOR ACC
MOVE:    STA     MOVSAV             ;SAVE THE ACC
MOVELP:  MOV     A,M                ;GET A SOURCE BYTE
         INX     H                  ;INCREMENT POINTER
         STAX    D                  ;PUT IN DESTINATION STRING
         ORA     A                  ;IS IT THE TERMINATOR?
         JZ      MOVOUT             ;YES, DONE
         INX     D                  ;INC DESTINATION POINTER
         JMP     MOVELP             ;KEEP MOVING
MOVOUT:  LDA     MOVSAV             ;RESTORE ACC
         RET                        ;RETURN
ENDALL=.
         .=0
          JMP  SPEAK
         NOP
         .WORD ENDALL,BUF
           .END
```

VITA

James A. Kutsch Jr. was born December 13, 1950 in Wheeling, W. Va. He graduated from West Virginia University with a A.B. in Psychology in 1972. He received his M.S. in Computer Science from West Virginia University in 1973, and received a Ph.D. in Computer Science from the University of Illinois in 1976.

While at West Virginia University, Dr. Kutsch was a systems analyst with the Electrical Engineering Department on a project to automatically monitor the environment of an operating coal mine. He also taught in the West Virginia University Computer Science and Statistics Department.

His publications include 'TASSY: One Approach to Individualized Instruction' which appeared in Proceedings of AFIPS Fall Joint Computer Conference 1972, and, 'SIMPDP11 and its Use in a Real-Time Monitoring System' which appeared in Proceedings of DECUS 1974.

Dr. Kutsch is a member of the Association for Computing Machinery, a fellow of the University of Illinois and member of Psy Chi and Phi Beta Kappa.

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-76-815	2.	3. Recipient's Accession No.
Title and Subtitle A Talking Computer Terminal			5. Report Date July 13, 1976 6.
Author(s) James Albert Kutsch, Jr.			8. Performing Organization Rept. No. UIUCDCS-R-76-815
Performing Organization Name and Address Department of Computer Science University of Illinois at Urbana-Champaign Urbana, Illinois 61801			10. Project/Task/Work Unit No. 11. Contract/Grant No.
Sponsoring Organization Name and Address Department of Computer Science University of Illinois at Urbana-Champaign Urbana, Illinois 61801			13. Type of Report & Period Covered Ph.D. Dissertation 14.

Supplementary Notes

Abstracts

Computer output is available to the blind only in a limited way. Braille listings, although easily available, are not the best answer. Other techniques, such as image conversion or speech synthesis are available only in experimental form. In this study, available means of providing computer output to the blind were reviewed and the criteria were set forth again which the effectiveness of a means of presentation could be measured. Available techniques were tested with these criterion and it was determined that a better method should be designed. As the result of this study, a talking computer terminal was constructed. The terminal is a computer independent, asychronous, ASCII dial-up device and can be connected to any computer allowing this type of communications. The hardware for the terminal consists of a speech synthesis device, a micro-processor, a keyboard, and an accoustic coupler. Software to provide the control of the speech synthesizer resides in the terminal's micro-processor system, permitting computer independence. The talking terminal was used extensively with the University of Illinois' PDP 10 computer and proved to be a very satisfactory means of comminications between a computer system and a blind computer scientist. It is hoped that the talking terminal will enable blind programmers to be more self-sufficient and productive, and will find many applications among sighted users.

17. Key Words and Document Analysis.

Talking Computer Terminal
Speech Synthesizer
Image Converters
Reading Machines

b. Identifiers/Open-Ended Terms

c. COSATI Field/Group

Availability Statement	19. Security Class (This Report)	21. No. of Pages
Release Unlimited	UNCLASSIFIED	91
	20. Security Class (This Page) UNCLASSIFIED	22. Price -----

FORM NTIS-35 (10-70) USCOMM-DC 40329-P71

NOV 1 5 1976

BOUND BY
HNM

UNIVERSITY OF ILLINOIS-URBANA
3 0112 039572828